Cauchy Interpolation for Multi-Variate and Multi-Derivative Data

Jonathan Kaufman

Master of Engineering

Department of Electrical Engineering

McGill University

Montréal,Québec

2007-05-24

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering

Your file Votre référence
ISBN: 978-0-494-38484-8
Our file Notre référence
ISBN: 978-0-494-38484-8

Dedication

To my late bubby, Becky.

Acknowledgments

I would like to thank first and foremost Professor J. Webb for his support and guidance without whom this Master's thesis would not be possible. I also would like to thank the following people for their support and guidance: Grégoire Guétat for help in debugging and getting the code to run correctly, Illan Kramer for help with debugging and LaTeX, Phil Marchand for support, guidance and for translating my abstract into french, Tracy Peppy for support and for always asking how it is going, and Josh Kobernick for pushing me when I needed a push. Finally, I would like to thank my mom and dad, who without their support this Master's thesis wouldn't be possible.

Abstract

There is often a need to interpolate data that is obtained through experiment or computational analysis, because the data is difficult or expensive to obtain. An example is the scattering parameters of microwave devices, obtained through computationally intensive finite element (FE) analysis. Cauchy interpolation is an established solution to this problem. In this thesis it is extended to interpolate data over a multi-parameter space, when the data available includes not just the function to be interpolated, but also its derivatives with respect to each parameter. The finite element method (FEM) provides such derivatives. The new algorithm is applied to a simple RLC circuit test case, and to real data from a 3D FE analysis of a rectangular waveguide component, in a 4-parameter space. Results show the effectiveness of the approach taken.

Abrégé

Il y a souvent un besoin d'interpoler des données obtenue à coût élevé d'expérimentation ou d'analyse informatique. Les paramètres de diffusions de dispositifs micro-onde, produit en utilisant de l'analyse par élément finie coûteuse, en est un exemple. L'interpolation Cauchy est une solution éprouvé à ce problème. Dans cette thèse, cette solution est augmenté pour interpolé des données sur un espace à plusieurs paramètres. Les données disponibles comprennent non seulement la fonction à être interpolé, mais aussi ses dérivés par rapport à chaque paramètres, fourni par la méthode à élément fini. Ce nouvel algorithme est appliqué à un simple jeu d'essai basé sur un circuit RLC, ainsi qu'à de vrai donnée obtenue de l'analyse à élément fini en 3D d'un guide d'ondes rectangulaire, formant un espace basé sur 4 paramètres. Les résultats démontre l'efficacité de l'algorithme utilisé.

Table of Contents

List of Tables

List of Figures

CHAPTER 1
Introduction

The ability to quickly and accurately solve complex electromagnetic problems is one of the primary goals of computational electromagnetics. As electromagnetic structures grow in complexity, so too does the amount of computational time required to effectively model these structures. While increases in computing power have been welcomed, it is not always effective to solve bigger and bigger problems by simply applying more computational power. Therefore an ongoing field of research in computational electromagnetics is focused on ways of finding accurate solutions quickly. The process of interpolating or extrapolating from a discrete data set to obtain accurate data over a continuous range is one such method of optimization. In this thesis, a method is proposed for taking a discrete set of finite element data and interpolating that data to produce accurate results over a large continuous range.

1.1 Objective

The scattering parameters of microwave devices are continuous functions of frequency and of various design parameters (e.g., geometric dimensions), and the microwave device designer is usually very interested in the nature of these functions. The problem is that experimental and computational methods can only provide the scattering parameters for one set of parameters at a time, e.g. for one fixed geometry. One could construct an almost continuous model by calculating the scattering parameters using a large set of fixed points, but this would prohibitively expensive.

Also, if these points fall along an area of significant change more points may be required to fully describe the scattering parameters, making this not a very efficient nor ideal method. Therefore there is a need for a continuous function to be constructed from these 'points' of data. If some discrete data set could be obtained and then interpolated to form a wide range continuous functional representation of the scattering parameter then it would be possible to provide the microwave engineer with a continuous function of scattering parameters of his/her microwave device at a very low computational cost.

The generation of a large range continuous model of complex microwave devices using the finite element method is a rather computationally expensive process. The objective of this thesis was to speed-up this process using Cauchy interpolation by only requiring the modeling of the microwave device at a few points rather than at many points and then interpolating the data from the finite element method and generating a large range continuous model of the device.

Current existing interpolation methods are only able to work for one unknown variable with higher order derivatives values of that parameter, or for multiple variables without any derivatives values of those parameters. The goal of this thesis is to extend the Cauchy interpolation method so as to be able to handle multiple variables with derivative information. Extending the Cauchy interpolation technique from a single variable to a multivariable technique will allow for the quick generation of continuous data models of microwave devices while decreasing the computational cost required to create these models. Therefore, the objective of this thesis is to

interpolate multivariate finite element data using Cauchy interpolation to provide the transfer function of microwave devices.

1.2 Literature Survey

This section presents an overview of the literature surveyed about using interpolation methods in the modeling of electromagnetic structures and their characteristics.

The performance of a microwave element at a single frequency is useful knowledge, but usually it is more desirable to know the performance of that microwave element over a whole range of frequencies. A very simplistic solution is to apply a single-frequency method repeatedly for a number of discrete frequencies over the desired range and then just combine them to get the performance characteristics over the desired range. An alternative approach as discussed in Webb [1] for computing the frequency response itself is: analysis at single frequency point, leading to the evaluation of higher derivatives with respect to the frequency at that point, followed by some form of high-order expansion of the quantity of interest over the whole frequency range. This basic idea is extended in this thesis to the simultaneous interpolation over both frequency and geometric parameters.

Rational polynomials have been used extensively to model frequency-domain responses [2]. An approach that is popular for rational polynomial is to use Padé approximations. This approach is based on using Taylor expansions of the parameter as a function frequency [2]. Sanaie *et al.* [3] used a Padé approximation with complex frequency hopping without poles such that the moments at different frequency points

were used to model the parameter of interest [2]. Celik *et al.* [4] used frequency-shifted moments to obtain the Padé approximation. Both [3] and [4] are examples of generating rational polynomials using Padé approximations.

The Cauchy method which is the focus of this thesis is an alternative to Padé approximations used to build rational polynomials. In [5] Adve and Sarkar examine the effects of noise in the data when the Cauchy method is applied to the system. Adve *et al.* extend this work on the Cauchy method with [2] where they present the Cauchy method as method for extrapolating wideband system response from a given narrow-band system response data. Kottapalli *et al.* demonstrated in [6] the use of Cauchy method with higher-order derivative terms to to compute the wide-band system response of a system using narrow-band data.The techniques demonstrated in [6] were extended by Adve et al in [2] where the coefficients were obtained directly using singular value decomposition. This allowed for the easy estimation of the order of the polynomial used in the rational model. Furthermore Adve *et al.* in [2] used the total least squares method to solve for the coefficients. This was advantageous because it allowed for the suppression of noise in the system.

The ability to solve for not just one variable but for multiple variables in a system at once is very powerful. In microwave waveguide devices, the frequency of the signal is just one of the parameters, there is also the parameters governing the geometry of the device. This creates a multidimensional problem. Lehmensiek *et al.* used an adaptive sampling algorithm for general multivariate interpolation based on Thiele-type branched continued fraction (BCF) representation of a rational function [7]. This technique constructs sets of single parameter interpolants at optimal points

in a (Dimension − 1) variable space. The univariate interpolants are then used to form bivariate interpolants which are then used to create trivariant interpolants, and so on [7]. This technique does not require derivatives [7], which means it cannot take advantage of derivative information that may already be available.

In the work of Lamecki *et al.* [8], a method of using multidimensional Cauchy method to create high quality multivariate models of microwave circuits from electromagnetic (EM) simulation data is discussed. Previous Cauchy methods allowed for a single variable, usually frequency, Lamecki *et al.* extended the concept to the multivariate case.

The goal of this thesis was to combine the work of Adve *et al.* [2] and Lamecki *et al.* [8] to create a method for solving multivariate systems with derivative information using the Cauchy Method. The method described uses the techniques of singular value decomposition (SVD) along with least squares (LS) and total least squares (TLS) to determine the order of the polynomials used in the rational equation and to solve for the coefficients. The inspiration for the total least squares technique is from Adve *et al.* [2].

CHAPTER 2
Theory

This chapter presents the mathematical theory that forms the groundwork for this thesis. The concepts of Cauchy interpolation, singular value decomposition (SVD), least squares and total least squares (TLS) are presented and explained.

2.1 Cauchy Interpolation

Cauchy interpolation is a method that provides accurate broadband information based on narrowband data. The focus of the Cauchy interpolation method in this thesis is on the rational transfer function $H(s)$ where s is the complex frequency[6]. The transfer function $H(s)$ can be characterized as a rational function which is represented by:

$$H(s) = \frac{A(s)}{B(s)} \tag{2.1}$$

which can be re-written in the form of (2.2):

$$A(s) - H(s)B(s) = 0 \tag{2.2}$$

where $A(s)$ and $B(s)$ are the sum of monomials terms:

$$A(s) = \sum_{k=0}^{p} a_k s^k \tag{2.3}$$

$$B(s) = \sum_{k=0}^{q} b_k s^k \tag{2.4}$$

The equations (2.2)-(2.4) after applying specific data points, where each row of the equation is a specific data point, can then be re-written in matrix form [2] with the unknowns a and b giving the following equations:

$$[A\vdots - HB]\begin{bmatrix} a \\ \\ b \end{bmatrix} = 0, \tag{2.5}$$

where the matrices $[A]$ and $[HB]$ are of the order of $N \times (p+1)$ and $N \times (q+1)$, respectively, and N is the number of data entries provided to the system.

2.1.1 Extending the Cauchy method to handle derivatives of the transfer function

The previous section showed the application of the Cauchy method with only information provided for a single variable and with no derivative information provided with respect to that single variable. The extension of the Cauchy interpolation method so as to handle derivatives of the transfer function is beneficial since the derivative information of a single point can be obtained if finite-element analysis is used. If we have the point s_j which represents the complex frequency then n derivatives of that single point can be calculated from the finite-element data at that single point[1]. With finite-elements it is possible to generate derivative information of the complex frequency variable and therefore a good idea to take advantage of this data.

This means that we can input to the system, for each s_j, not just the value of the transfer function, but the values of all its derivatives up to the n^{th}. This is an advantageous feature since with finite element method (FEM) data it is rather computationally expensive to generate data per variable point, but for each point

the higher-order derivatives are relatively cheap computationally to generate. Differentiating equation 2.2 n times with respect to s and then evaluating all the terms at the frequency point s_j gives the following equation [9]:

$$A^{(n)}(s_j) = \sum_{i=0}^{n} \binom{n}{i} \left(\frac{\partial^{(i)} H}{\partial x_0^{(i)}} \bigg|_{\underline{s}_j} \right) \frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} B \bigg|_{\underline{s}_j} \tag{2.6}$$

Here $\binom{n}{i}$ is:

$$\frac{n!}{i!(n-i)!} \tag{2.7}$$

2.1.2 Extending the Cauchy method into multi-variate space

The Cauchy method started off as a method for interpolating from narrowband data to broadband data. Now a useful extension of this concept would be to extend this to multi-variable data. This would mean that the data space that the Cauchy method is applied to is multi-dimensional. Equations (2.3) and (2.4) can be rewritten for the multi-variable case:

$$A(\underline{x}) = \sum_{k_0=0}^{p} \sum_{k_1=0}^{p-k_0} \sum_{k_2=0}^{p-k_0-k_1} \cdots\cdots \sum_{k_G=0}^{p-k_0-k_1-k_2\cdots k_{G-1}} x_0^{k_0} x_1^{k_1} x_2^{k_2} \cdots x_G^{k_G} a_{k_0 k_1 k_2 \cdots k_G} \tag{2.8}$$

$$B(\underline{x}) = \sum_{k_0=0}^{q} \sum_{k_1=0}^{q-k_0} \sum_{k_2=0}^{q-k_0-k_1} \cdots\cdots \sum_{k_G=0}^{q-k_0-k_1-k_2\cdots k_{G-1}} x_0^{k_0} x_1^{k_1} x_2^{k_2} \cdots x_G^{k_G} b_{k_0 k_1 k_2 \cdots k_G} \tag{2.9}$$

where the x_0 term in both (2.8) and (2.9) is the complex frequency term s of (2.3) and (2.4). The terms x_1, x_2 until x_G in (2.8) and (2.9) represent the other parameters

of the problem (e.g. the geometric parameters of a device being analyzed by the FEM), where p and q are the degree of the polynomial. Therefore, we now have a way to represent the multi-variable Cauchy method. The next step is to combine the multi-derivative case with the multi-variable case.

2.1.3 The multi-derivative, multi-variable case

Now combining the multi-variable and the multi-derivative cases will give the the multi-derivative, multi-variate case. The combination of (2.6) with (2.8) and (2.9) gives the following equations (2.10) and (2.11):

$$\frac{\partial^{(n)}}{\partial x_0^{(n)}} A\Big|_{\underline{x}_j} - \sum_{i=0}^{n} \binom{n}{i} \left(\frac{\partial^{(i)} H}{\partial x_0^{(i)}}\Big|_{\underline{x}_j} \right) \frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} B\Big|_{\underline{x}_j} = 0 \tag{2.10}$$

$$\begin{aligned} \frac{\partial}{\partial x_m}\frac{\partial^{(n)}}{\partial x_0^{(n)}} A\Big|_{\underline{x}_j} - \sum_{i=0}^{n} \binom{n}{i} \Bigg\{ \left(\frac{\partial}{\partial x_m}\frac{\partial^{(i)} H}{\partial x_0^{(i)}}\Big|_{\underline{x}_j} \right) \frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} B\Big|_{\underline{x}_j} \\ + \left(\frac{\partial^{(i)} H}{\partial x_0^{(i)}}\Big|_{\underline{x}_j} \right) \left(\frac{\partial}{\partial x_m}\frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} B \right)\Big|_{\underline{x}_j} \Bigg\} = 0 \end{aligned} \tag{2.11}$$

The first equation (2.10) is obtained by differentiating A=HB with respect to x_0 (n times). Equation (2.11) is obtained by along with differentiating once with respect to x_1 through to x_G. In principle, more derivatives with respect to $x_1 \ldots x_G$ could be taken, but the FEM used provides just the first derivative the geometric parameters. Equations (2.10) and (2.11) lead to a matrix system.

$$[A \vdots B] \begin{bmatrix} a \\ b \end{bmatrix} = 0 \tag{2.12}$$

A is an $N \times (P+1)$ matrix and B is an $N \times (Q+1)$ matrix, where:

- N is the total number of equations of the form (2.10) or (2.11);
- $P+1$ is the number of unknowns coefficients in a in (2.8);
- $Q+1$ is the number of unknowns coefficients in b in (2.9).

2.1.4 Choosing N, P and Q

The matrix has N rows and M columns.

1. $N < M-1$. The rank R is at most $M-2$. The number of null vectors $(M-R)$ is greater than 1, meaning that there is more than one interpolation that exactly fits the data. This is unnecessary and undesirable in general, so we should either increase N or decrease P and Q to avoid it.
2. $N \geq M-1$. Two cases:
    a. The finite-element (FE) case, or any case where the actual transfer function is not a rational function. Assume rank R at least $M-1$. (If the rank is less than $M-1$, which could happen if not enough independent data points are provided, then we are back to the case in 1. Increase N until $R = M-1$ or $R = M$.)
        i. If $R = M$, then there are no null vectors, i.e. no exact solution to the matrix problem, which means there is no solution that matches the N data points exactly. However, we can get some form of the solution that tries to match the data as well as possible (e.g. TLS).

ii. If $R = M - 1$, then there is one null vector and this null vector is the exact, and unique solution to the matrix problem. The null vector represents an interpolation that fits the N data points exactly. Although this is one way of handling the FE case, it is probably better to decrease P and Q (hence decrease M) and use a TLS fit.

b. The RLC case (a test case consisting of a circuit made up of a resistor, capacitor and inductor), or any case where the actual transfer function is a rational function for some values of P and Q, say $P = P_0$ and $Q = Q_0$. Again, there are two subcases:

i. $P < P_0$ or $Q < Q_0$.This is just like the FE case, above, because the interpolation does not have high enough P or Q to represent the exact transfer function.

ii. $P \geq P_0$ and $Q \geq Q_0$. The rank is at most $M - 1$. There is at least one null vector. Each null vector represents an interpolation that fits the N data points exactly. When $P = P_0$ and $Q = Q_0$ and N is big enough (i.e., there are enough independent data points), the rank is $M - 1$ and there is just one null vector. This is the most desirable solution for this type of problem (and it is what is given in (2.23)). Assuming all the data points are independent, $N = M-1$ is sufficient.

2.2 Singular Value Decomposition

Orthogonality is a very important concept in linear algebra and matrix computations [9]. Singular value decomposition relies heavily on the concept of orthogonality. A matrix A which is $n \times n$ is said to be orthogonal if that matrix multiplied by its

complex transpose is equal to the identity matrix:

$$AA^H = I \tag{2.13}$$

A matrix that is orthogonal also means that it is always invertible which means that $A^{-1} = A^T$ [10]. Singular value decomposition (SVD) exploits the concept of orthogonality it can be used to determine the rank of a matrix and also can detect the presence of a null-vector or vectors in a matrix.

2.2.1 Theory behind singular value decomposition

If A is a matrix of the form $n \times m$ where $n \geq m$, then the matrix A is said to be *overdetermined* [11]. Then, if matrix A is *overdetermined*, it can be de-constructed using the singular value decomposition [9]. Applying singular value decomposition to matrix A then results in the following (2.14) [12]:

$$[U]\,[\Sigma]\,[V]^H = \mathrm{SVD}\,(A) \tag{2.14}$$

where:

$$U = [u_1, \cdots, u_m] \in \mathbb{R}^{m \times m} \tag{2.15}$$

$$V = [v_1, \cdots, v_n] \in \mathbb{R}^{n \times n} \tag{2.16}$$

The matrices U and V are unitary which that means that the hermitian (or the *complex-conjugate transpose*) of the matrix U is equal to the inverse of matrix U, and because U and V are square and non-singular then the matrix U multiplied by its inverse is equal to its identity matrix [10]:

$$U^H \;=\; U^{-1} \tag{2.17}$$

$$UU^{-1} = I \tag{2.18}$$

Also:

$$\begin{aligned} \Sigma &= diag(\sigma_1, \cdots, \sigma_p) \in \mathbb{R}^{m \times n}, \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0 \\ where\ p &= min\{m, n\} \end{aligned} \tag{2.19}$$

Σ has the form:

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & & \mathrm{O} \\ & \sigma_2 & \\ & \ddots & \\ \mathrm{O} & & \sigma_p \\ \cdots\cdots & \cdots\cdots & \cdots\cdots \\ & \mathrm{O} & \end{bmatrix} \tag{2.20}$$

2.2.2 Using SVD to find the rank of a matrix

Singular value decomposition (SVD) is a valuable tool in determining the rank of a matrix. The σ_i elements are the diagonal values in Σ and are the singular values of the matrix. These values can be used to calculate the rank. This is very important in terms of determining the solution vector for our problem. Because of the number of non-zero values in the matrix in equation (2.5) gives the rank of the matrix [2]. Though when determining the actual rank of the matrix, a threshold must be set for what constitutes a zero due to the presence of floating point error which can cause what should be a zero to be a very small number, therefore the term that is regarded as zero must be a certain threshold (orders of magnitude) smaller than the previous

non-zero singular value.

$$\overbrace{N\{[A\vdots B]}^{\text{M}} \tag{2.21}$$

M is size of the column space of A, and therefore the rank of A added with the null vector or vectors is equal to the complete column space: $Rank(A)+nullvector(s) = M$. The power of the monomial terms of the Cauchy matrix will determine the number of columns and thus the rank of the matrix. The values for P and Q from equation (2.8) and equation (2.9) determine M according to the following equation:

$$M = P + Q + 2 \tag{2.22}$$

The value of the $Rank(A)$ therefore has to be one less than that for M because of the need to have exactly 1 null vector, therefore equation (2.22) can be re-written as:

$$M = Rank(A) + 1 = P + Q + 2 \tag{2.23}$$

Therefore in order to solve the problem, the rank of the matrix must be one less than the column space and also 1 column vector therefore must be the null vector of matrix A [11].

The rank of a matrix is the number of independent rows or columns of a matrix. The null vectors are the columns or rows where if we have a matrix Z then if there is a vector v where $Zv = 0$ then the vector v is a null vector [11]. It is the null vector that is the key to the uniqueness of the solution, there must be exactly 1 null vector for there to be a unique solution. The presence of a null vector is detected using the singular value decomposition if the last term along the diagonal of Σ is 0, then the

rank of A is one less than the matrix space and therefore there exists 1 null vector and thus a unique solution vector. If there is no null vector and the rank is equal to the amount of columns in the matrix, then there is no solution. If there are 2 or more null vectors, then there is not one unique solution but, in fact, an infinite number of solutions.

2.2.3 Using the SVD to find the solution

Therefore we can use the results from the singular value decomposition to find the unknowns $a_{0\ldots P}$ and $b_{0\ldots Q}$. With (2.20) the following equation can be derived:

$$AV = U\Sigma V^H V \tag{2.24}$$

Which is equal to:

$$AV = U\Sigma \quad (since\ V^H = V^{-1}) \tag{2.25}$$

So $Av_p = \sigma_p u_p$ where v_p is the p^{th} column of V. And where u_p is the p^{th} column of U. If $\sigma_p = 0$, $Av_p = 0$ and thus v_p is a null vector of A and is a solution vector for the a and b terms of equation (2.5). The actual answer itself is only defined to within a scale multiple. In other words, if v_1 is one "actual answer", then $2 * v_1$ is an equally good "actual answer" [2].

2.3 The Least Squares and Total Least Squares Methods

Unfortunately, in many problems an exact solution is impossible, therefore a method of fitting is needed. Two methods were different methods were used in this thesis, least squares (LS) and total least squares (TLS). Both methods can be used to find an approximate solution. For any finite set of data, it is possible to get an exact answer (i.e. an exact fit of the data by a rational function A/B) using some

sufficiently large P and Q values. This would be true even in the finite-element case. However, we can distinguish the cases better in this way:

1. Cases in which the data are coming from an underlying function H which is actually a rational function. One example is the transfer function $H = \frac{V_{out}}{V_{in}}$ for a RLC circuit. In this case, there will be a P and Q that can be selected such that we can exactly fit the data no matter how many data points are given.

2. Cases in which the data are coming from an underlying function H which is NOT a rational function (e.g. from FE analysis). In this case, there is no finite P and Q that will give an exact fit for all data sets. However, in practice the dominant behavior of H can be captured by choosing "reasonable" values of P and Q and then doing a least squares fit to the data. The fitting also, of course, has the advantage that it tends to ignore the numerical "noise" coming from a finite-element analysis.

Total least squares is a computational method which is very similar to the least squares (LS) method but with certain fundamental differences which will be detailed in later sections. One of the reasons that some systems have no exact solution is because the input data is corrupted by noise and errors. Therefore the least squares (LS) and total least squares (TLS) methods are powerful methods that can be used to minimize the effect of noise on these systems while providing the best, most accurate possible solution.

2.3.1 Basic theory behind total least squares and least squares

Least squares (LS) and total least squares (TLS) are methods for solving *overdetermined* sets of the linear equation $Ax \approx b$ [12]. An *overdetermined* set is when there are more equations than unknowns [11]. In the situation of *overdetermined*

equations, no exact solution is possible and furthermore there is the presence of noise (or error) in the input data. Both TLS and LS methods are methods of fitting the results to get the best possible solution. To understand both total least squares and least squares a good way to start is to examine the least squares concept first. LS is where the goal is to find the vector $x \in \mathbb{R}^n$ such that $Ax \approx b$ where the matrix A is of the form $n \times m$ and where $n \geq m$ meaning that the matrix is *overdetermined* and the observation vector $b \in \mathbb{R}^m$ is given [12]. Starting with the initial problem of:

$$Ax \approx b, \tag{2.26}$$

the LS method seeks to minimize equation (2.27):

$$minimize \, \|Ax - b\|_2 \qquad A \in \mathbb{R}^{m \times n}, \quad b \in \mathbb{R}^m, x \in \mathbb{R}^n \tag{2.27}$$

So the goal of LS and also TLS, is to *minimize* the quadratic in (2.27) [13]. In our case the equation (2.27) can be re-written in the form of:

$$minimize \, \|Ax - 0\|_2 \tag{2.28}$$

Now if the problem is ideal, which means there is no experimental error nor floating point error introduced into the floating point calculation, and P and Q are big enough so that A/B is an exact fit to the given data, then equation (2.28) should give zero. But since we do have experimental and floating point error and no exact fit is possible then we must seek to *minimize* (2.28) using TLS. The basic concept behind TLS is: given an *overdetermined* set of m linear equations $Ax \approx b$ in n

unknowns, the TLS method will [12]:

$$minimize \left\| [A;b] - \left[\widehat{A};\widehat{b}\right] \right\|_F \quad where \left[\widehat{A};\widehat{b}\right] \in \mathbb{R}^{m\times(n+1)} \tag{2.29}$$

$$subject\ to\ \widehat{b}\ belonging\ to\ the\ range\ of \widehat{A} \tag{2.30}$$

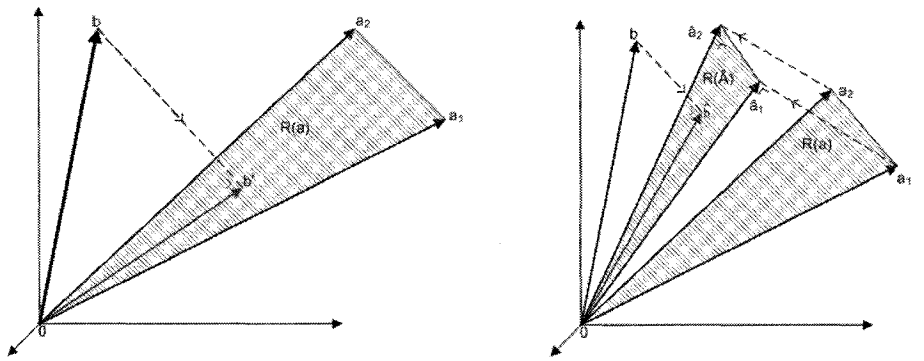
Once a minimizing $\left[\widehat{A};\widehat{b}\right]$ is found, then any x satisfying the following equation (2.31)

$$\widehat{A}x = \widehat{b} \tag{2.31}$$

is called a total least squares solution and $\left[\Delta\widehat{A};\Delta\widehat{b}\right] = [A;b] - \left[\widehat{A};\widehat{b}\right]$ the corresponding total least squares correction. The $\Delta\widehat{A}$ and $\Delta\widehat{b}$ represent the error in vectors, it is assumed that the error is an even distribution [12]. The difference between the LS method and the TLS method is that in LS method, all the "error" is assumed to be in the *RHS*, b, and the matrix A is assumed to be exactly known. In TLS, both A and b are assumed to be in error. Shown in Figure 2–1 is a graphical representation of a simple problem using the LS method and the total least squares method. The advantageous in accuracy of the TLS method over the LS method can be seen in the figure.

2.3.2 Least squares and total least squares and the null vector

Both least squares techniques, the regular least squares (LS) and the total least squares (TLS) relies heavily on the results given by the singular value decomposition (SVD) of the A matrix. If the SVD of A results in $\sigma_{n+1} \neq 0$ then the rank of A is $n+1$ and therefore there is no non-zero vector in the orthogonal complement of the space, therefore there is no exact solution to the equation $Ax = b$ but an



(a) The least squares solution by projecting b orthogonally onto R(A) and solving $Ax = b'$ [12]

(b) The total least squares solution is obtained by approximating the columns a_i of A and b by $\widehat{a_i}$ and $\widehat{b}$ until $\widehat{b}$ is in the space $R\left(\widehat{A}\right)$, generated by the columns $\widehat{a_i}$ and $\widehat{A}x = \widehat{b}$ [12]

Figure 2–1: Graphical representation of LS and TLS

approximate solution can be obtained for $Ax \approx b$ by using either the TLS method or the LS method [12]. But if $\sigma_{n+1} = 0$, an exact solution is obtainable and TLS is not needed. Nevertheless, this case serves as a good way to test an implementation of TLS or LS, which should both give the same answer as obtained from SVD.

2.3.3 Applying least squares methods to solve multi-dimensional vector problems

In the case where the input data are finite-Element (FE) data or experimental data and therefore the transfer function of the system cannot be known beforehand, the least squares or total least squares method can be used to solve for the transfer function. In the case that the $\sigma_p \neq 0$ (see equation (2.25)) due to the presence of errors or inaccuracies in the input data and because it is not possible to find values of P and Q that would provide an exact fit, applying basic LS to try and solve this

problem will not result in satisfactory values but TLS will. The method for applying TLS to multi-dimensional vector problems is slightly more complicated because the errors are present in the $-HB$ portion of the matrix of data and there are no errors in the A portion of the matrix. Therefore, to use TLS on multi-dimensional vector problems, orthogonal projection of the matrix is taken. So QR decomposition is applied to the matrix $[A - HB]$ to obtain these orthogonal projections [14]. The QR decomposition of the matrix results in a Q matrix and a R matrix. The Q matrix is an *orthogonal matrix* thus it is a matrix that satisfies the following equation (2.32) [10]

$$Q^T Q = I \qquad \textit{where } I \textit{ is the identity matrix} \tag{2.32}$$

And where the resulting R matrix is an *upper triangular matrix* of the following form (2.33) [2]:

$$\begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = 0 \tag{2.33}$$

The system to solve the unknown equation $\begin{bmatrix} a \\ b \end{bmatrix}$ is now shown in (2.34):

$$R \begin{bmatrix} a \\ b \end{bmatrix} = 0 \tag{2.34}$$

Where the R matrix which results from the QR decomposition of the matrix can be divided by the following equation (2.35)

$$\begin{array}{cc} \overbrace{\phantom{R_{11}}}^{\text{P+1}} & \overbrace{\phantom{R_{12}}}^{\text{Q+1}} \\ \left[\begin{array}{cc} R_{11} & R_{12} \\ 0 & R_{22} \end{array}\right] & \begin{array}{l} \} \ P+1 \\ \} \ N-(P+1) \end{array} \end{array} \tag{2.35}$$

Where $P+1$ and $Q+1$ is the total number of coefficients in A and in $-HB$ portions of the Cauchy matrix respectively and where N is the number of rows in the Cauchy matrix.

The next step in solving the system is to take the singular value decomposition of the sub-matrix R_{22} from the R matrix. This gives:

$$[U]\,[\Sigma]\,[V^H]\,b = 0 \tag{2.36}$$

The least squares solution is the last column of the matrix V:

$$b = [V]_{Q+1} \tag{2.37}$$

Now to solve for the unknowns a we need to solve the following equation (2.38) [2]

$$a = -R_{11}^{-1} R_{12} b \tag{2.38}$$

CHAPTER 3
Implementation

This chapter shows the connection between the theory outlined in the previous chapter and how this theory was applied. Also, outlined in this chapter, are some of the tools and techniques used to create the Cauchy interpolation application.

3.1 Development Environment

Programming for this thesis was done using the MATLAB 7 software package. The MATLAB software was selected due to a number of advantageous features, such as the MATLAB scripting language, which is specifically designed for mathematical problems. The MATLAB scripting language allows for the easy creation of vectors and matrices without the need to worry about memory allocation, and the ability to create variables without the need to typecast the variables as either integer or float type. Also the MATLAB software package comes with numerous integrated math functions, and one of these built-in functions that was extremely useful to this thesis was the singular value decomposition (SVD) function. The MATLAB software also contained numerous post-processing features, which allowed for the easy creation of detailed graphs of the obtained results.

3.2 The Organization Of The Code

The code was organized and written in a modular fashion, this was done in order to make the debugging process easier. The Cauchy interpolation application code is organized into five main sections. The primary section contains all the initial

parameter settings, reads the input data file, and then handles the control of the program while sending and receiving data through function calls to the other sections of the application. The four other sections build parts of the A and B matrices. The primary section assembles these parts into the system matrix, $[A|B]$, and uses the total least squares method to find the vectors a and b.

3.3 Converting Theoretical To Practical

As will be seen in subsection 3.3.1, there are three main *for* loops, that are implemented in the Cauchy interpolation application. The *for* loops are used to control how the input data is processed by the Cauchy application. The first *for* loop, which is the outermost loop, loads in the data vector $\underline{x}_j$. It runs once, then all the code contained in it executes and then afterwards a new data vector $\underline{x}_j$ is loaded in. The next line in the pseudo-code is another *for* loop. This *for* loop goes from $n = 0$ to D, where the value of n represents the current power of the derivative term that the system is operating on and D is the highest power derivative, that the program will be given. There is one last *for* loop and this loop is inside the other two loops. This *for* loop only runs when the geometric variables are being used in the application. Therefore when the application is processing the multi-variable case, this loop runs on the geometric variables that go from x_1 to x_G, where x_G is the total number of geometric terms in the system. But if the application is processing an example with only a frequency variable x_0 and no geometric variables then this loop and the code contained in this loop will not execute.

3.3.1 Pseudo-code of the Cauchy application main

The pseudo-code shown below is a pseudo-code representation of the code used in the construction of the A and B matrices.

for $j = 1$ to J **do**

for $n = 0$ to D **do**

$$\frac{\partial^{(n)}}{\partial x_0^{(n)}} A \Big|_{\underline{x}_j} - \sum_{i=0}^{n} \binom{n}{i} \left(\frac{\partial^{(i)} H}{\partial x_0^{(i)}} \Big|_{\underline{x}_j} \right) \frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} B \Big|_{\underline{x}_j} = 0$$

for $m = 1$ to G **do**

$$\frac{\partial}{\partial x_m} \frac{\partial^{(n)}}{\partial x_0^{(n)}} A \Big|_{\underline{x}_j} - \sum_{i=0}^{n} \binom{n}{i} \left\{ \left(\frac{\partial}{\partial x_m} \frac{\partial^{(i)} H}{\partial x_0^{(i)}} \Big|_{\underline{x}_j} \right) \frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} B \Big|_{\underline{x}_j} \right.$$

$$\left. + \left(\frac{\partial^{(i)} H}{\partial x_0^{(i)}} \Big|_{\underline{x}_j} \right) \left(\frac{\partial}{\partial x_m} \frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} B \right) \Big|_{\underline{x}_j} \right\} = 0$$

end for

end for

end for

3.3.2 Analysis of the pseudo-code

After the first two *for* loops shown in subsection 3.3.1, the next line is:

$$\frac{\partial^{(n)}}{\partial x_0^{(n)}} A \Big|_{\underline{x}_j} - \sum_{i=0}^{n} \binom{n}{i} \left(\frac{\partial^{(i)} H}{\partial x_0^{(i)}} \Big|_{\underline{x}_j} \right) \frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} B \Big|_{\underline{x}_j} = 0 \tag{3.1}$$

Shown in (3.1) is the single variable with unlimited derivatives implementation of equation (2.5). The complex frequency parameter x_0 can have an unlimited number of derivatives. The other parameters x_1 through x_G are assumed to only have one derivative term associated with them. The equation (3.1) can be broken down and analyzed. The term:

$$\frac{\partial^{(n)}}{\partial x_0^{(n)}} A \Big|_{\underline{x}_j} \tag{3.2}$$

in equation (3.2) can be re-written by substituting (2.8):

$$\frac{\partial^{(n)}}{\partial x_0^{(n)}} A \Big|_{\underline{x}_j} = \left(\sum_{k_0=0}^{p} \sum_{k_1=0}^{p-k_0} \cdots \sum_{k_G=0}^{p-k_0-k_1 \ldots k_{G-1}} \frac{\partial^{(n)}}{\partial x_0^{(n)}} x_0^{k_0} x_1^{k_1} \ldots x_G^{k_G} a_{k_0 k_1 k_2 \cdots k_G} \right) \Bigg|_{\underline{x}_j} \tag{3.3}$$

The values for variables x_1 to x_G are read in from the input data file, and taken to the power of k_0 to k_G, respectively. Now $\frac{\partial^{(n)}}{\partial x_0^{(n)}} x_0^{k_0}$ can be written as:

$$k_0 \left(k_0 - 1\right) \ldots \left(k_0 - n\right) x_0^{k_0 - n} \tag{3.4}$$

The above equation (3.4) is for the general case but it needs to be defined for the following special cases that can arise:

$$\frac{\partial^{(n)}}{\partial x_0^{(n)}} x_0^{k_0} \begin{cases} k_0 \left(k_0 - 1\right) \ldots \left(k_0 - n\right) x_0^{k_0 - n} & k_0 \geq n \\ 0 & k_0 < n \\ 1 & k_0 = n = 0 \end{cases} \tag{3.5}$$

Again looking at equation (3.3), depending on the number of geometric parameters given to the system, the number of $\sum$ summations and the limits of those $\sum$ summations changes. This means that the system must dynamically handle the

creation and the limits of these $\sum$ summations. This dynamic creation of $\sum$ summations and their limits will be described later in section 3.4.

The second part of (3.1) represents the HB part of the $A - HB$ equation and can be re-written as shown in (3.6):

$$\sum_{i=0}^{n} \binom{n}{i} \left(\frac{\partial^{(i)} H}{\partial x_0^{(i)}} \bigg|_{\underline{x}_j} \right) \frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} B \bigg|_{\underline{x}_j} =$$

$$\sum_{i=0}^{n} \binom{n}{i} \left(\frac{\partial^{(i)} H}{\partial x_0^{(i)}} \bigg|_{\underline{x}_j} \right) \frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} \left(\sum_{k_0=0}^{q} \sum_{k_1=0}^{q-k_0} \cdots \sum_{k_G=0}^{q-k_0-k_1} x_0^{k_0} x_1^{k_1} \ldots x_G^{k_G} b_{k_0 k_1 k_2 \cdots k_G} \right) \bigg|_{\underline{x}_j} \tag{3.6}$$

The right hand side of (3.6) can be re-written as:

$$= \sum_{k_0=0}^{q} \sum_{k_1=0}^{q-k_0} \cdots \sum_{k_G=0}^{q-k_0-k_1-k_G} \left[\sum_{i=0}^{n} \binom{n}{i} \frac{\partial^{(i)} H}{\partial x_0^{(i)}} \bigg|_{\underline{x}_j} \left(\frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} x_0^{k_0} x_1^{k_1} \ldots x_G^{k_G} \bigg|_{\underline{x}_j} \right) b_{k_0 k_1 k_2 \cdots k_G} \right] \tag{3.7}$$

There are two separate derivative terms in (3.7), the first term $\left. \frac{\partial^{(i)} H}{\partial x_0^{(n-i)}} \right|_{\underline{x}_j}$ is the derivative of the transfer function H and this value is input into the system from the input data file. Details regarding the input file will follow in section 3.6. The second part $\frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} x_0^{k_0} x_1^{k_1} \ldots x_G^{k_G}$ is handled using equation (3.5).

3.3.3 Handling the multi-variable case

Now after the equation obtained by differentiating $A = HB$ with respect to x_0 (n times) shown in section 3.3.1, there is another *for* loop. This *for* loop depends on which geometric parameter the system is operating on. The complex frequency

variable is handled separately from all the other variables because it has unlimited derivatives associated with it while the second part is for the geometric parameters only have a single derivative associated with them.

$$\frac{\partial}{\partial x_m}\frac{\partial^{(n)}}{\partial x_0^{(n)}}A\bigg|_{\underline{x}_j} - \sum_{i=0}^{n}\binom{n}{i}\left\{\left(\frac{\partial}{\partial x_m}\frac{\partial^{(i)}H}{\partial x_0^{(i)}}\bigg|_{\underline{x}_j}\right)\frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}}B\bigg|_{\underline{x}_j}\right. \tag{3.8}$$
$$\left. + \left(\frac{\partial^{(i)}H}{\partial x_0^{(i)}}\bigg|_{\underline{x}_j}\right)\left(\frac{\partial}{\partial x_m}\frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}}B\right)\bigg|_{\underline{x}_j}\right\} = 0$$

Now equation (3.8) can be divided into two parts. The first part is everything to the left of the minus sign and the second part is everything to the right of the minus sign. Each part was implemented separately, to make the code easier to handle and debug. The first part is:

$$\frac{\partial}{\partial x_m}\frac{\partial^{(n)}}{\partial x_0^{(n)}}A\bigg|_{\underline{x}_j} \tag{3.9}$$

Now (3.9) expands to:

$$\frac{\partial}{\partial x_m}\frac{\partial^{(n)}}{\partial x_0^{(n)}}A\bigg|_{\underline{x}_j} = \frac{\partial}{\partial x_m}\frac{\partial^{(n)}}{\partial x_0^{(n)}}\bigg|_{\underline{x}_j}\left(\sum_{k_0=0}^{p}\sum_{k_1=0}^{p-k_0}\cdots\sum_{k_G=0}^{p-k_0-k_1-k_G} x_1^{k_1}x_0^{k_0}\ldots {x_G}^{k_G}a_{k_0k_1k_2\cdots k_G}\right)\bigg|_{\underline{x}_j} \tag{3.10}$$

which can be re-written again as

$$\frac{\partial}{\partial x_m}\frac{\partial^{(n)}}{\partial x_0^{(n)}}A\bigg|_{\underline{x}_j} = \left(\sum_{k_0=0}^{p}\sum_{k_1=0}^{p-k_0}\cdots\sum_{k_G=0}^{p-k_0-k_1-k_G}\frac{\partial^{(n)}x_0^{k_0}}{\partial x_0^{(n)}}x_1^{k_1}\cdots\frac{\partial x_m^{k_m}}{\partial x_m}\cdots x_G^{k_G}\right)\bigg|_{\underline{x}_j} a_{k_0k_1k_2\cdots k_G} \tag{3.11}$$

The term $\frac{\partial x_m^{k_m}}{\partial x_m}$ which is applied to x_1 to x_G can be re-written as:

$$\frac{\partial x_m^{k_m}}{\partial x_m} = k_m x_m^{k_m - 1} \tag{3.12}$$

Equation (3.12) was implemented as shown below in (3.13):

$$\frac{\partial}{\partial x_m} x_m^{k_m} = \begin{cases} k_m x_m^{k_m-1} & k_m > 1 \\ 1 & k_m = 1 \\ 0 & k_m = 0 \end{cases} \tag{3.13}$$

Now $\frac{\partial^{(n)}}{\partial x_0^{(n)}}$ is handled separately and is the same as shown in (3.5).

The next part with the two terms involving B (3.8) are handled in the same way as the terms involving A:

$$(-1)\sum_{i=0}^{n} \binom{n}{i} \left\{ \left(\frac{\partial}{\partial x_k} \frac{\partial^{(i)} H}{\partial x_0^{(i)}} \bigg|_{\underline{x}_j} \right) \frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} B \bigg|_{\underline{x}_j} + \left(\frac{\partial^{(i)} H}{\partial x_0^{(i)}} \bigg|_{\underline{x}_j} \right) \left(\frac{\partial}{\partial x_k} \frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} B \right) \bigg|_{\underline{x}_j} \right\} = 0 \tag{3.14}$$

In equation (3.14), the part $\left(\frac{\partial}{\partial x_k} \frac{\partial^{(i)} H}{\partial x_0^{(i)}} \right) \Big|_{\underline{x}_j}$ represents the geometric derivative applied to the frequency derivative. The values for $\left(\frac{\partial}{\partial x_k} \frac{\partial^{(i)} H}{\partial x_0^{(i)}} \right) \Big|_{\underline{x}_j}$ are pre-generated and the application takes them from the input data file. The two terms involving B in (3.14) are handled in the same way as the terms involving A. The next part is $\frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}} B \Big|_{\underline{x}_j}$ which can be re-written as:

$$\frac{k_0!}{(k_0 - (n-i))!} x^{k_0 - (n-i)} x_1^{k_1} \ldots x_G^{k_G} \tag{3.15}$$

Then the next part is $\left.\frac{\partial^{(i)} H}{\partial x_0^{(i)}}\right|_{\underline{x}_j}$, this represents the derivatives of the transfer function H and this data is inputted into the system from the input data file. The last part is $\frac{\partial}{\partial x_m}\frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}}B\Big|_{\underline{x}_j}$ which can be broken into two parts where $\frac{\partial^{(n-i)}}{\partial x_0^{(n-i)}}$ can be re-written as equation (3.15) and the part $\frac{\partial}{\partial x_m}$ is applied to the variables x_1 through x_G and can be re-written as:

$$\frac{\partial}{\partial x_m} = k_m x_m^{k_m - 1} \tag{3.16}$$

Also equation (3.16) is subject to the same conditions as shown in equation (3.13).

3.4 Using Recursion to Generate For Loops

The Cauchy interpolation system required the implementation of dynamic *for* loops. Because the number of loops required and the limits of those loops change depending on the values of p and q and on the number of geometric parameters entered into the system, the number of *for* loops required had to be determined at run-time. Therefore a method was needed to generate the appropriate number of *for* loops at run-time to implement:

$$\sum_{k_0=0}^{p}\sum_{k_1=0}^{p-k_0}\cdots\sum_{k_G=0}^{p-k_0-k_1} \quad \text{and} \quad \sum_{k_0=0}^{q}\sum_{k_1=0}^{q-k_0}\cdots\sum_{k_G=0}^{q-k_0-k_1} \tag{3.17}$$

The limits of the summations are determined by the variables p and q, where p and q are the highest powers of the monomial terms. The values of p and q therefore control the number of columns $P+1$ and $Q+1$ that are created in the matrix as shown in (2.22). The relationship between the limits of the summations p and q and the numbers P and Q can be derived as follows:

If we have G+1 variables $x_0, x_1, \ldots, x_G$, then the number of monomials of degree $\leq p$ is equal to P. Therefore the following can be written:

$$x_0^{k_0} x_1^{k_1} \ldots x_G^{k_G} \text{ with } 0 \leq k_0 + k_0 \ldots + k_G \leq p \tag{3.18}$$

Then let $f(i,j) = \textit{number of monomials of degree} \leq i \text{ in } j$ variables. We can now formulate a recursive formula for P:

$$P = f(p, G+1) = \sum_{k_0=0}^{p} f(p - k_0, G) \qquad p \leq 0, \quad G \leq 0 \tag{3.19}$$

$$\textit{with } f(i,1) = i + 1 \qquad i \geq 0 \tag{3.20}$$

The values of p and q are selected in such a way, so that the Cauchy interpolation matrix is a $M \times N$ matrix where $M \geq N$. For example, if there are three variables: one complex frequency variable and two geometric variables then three *for* loops are required. However, if there are five variables (one complex frequency variable and four geometric variables) then five *for* loops are required, furthermore the loops do not all have the same limits. Therefore, recursion was selected as the method of implementing this part.

The advantages of recursion is that allows for dynamic creation of *for* loops though there are a few critical things to be aware of when using recursion. One disadvantage recursion suffers from is the danger of *stack-overflow*, *stack-overflow* occurs when the instruction stack that contains all the recursive copies continually grows until it reaches its limit causing the program to crash this is usually caused by either an error in the code causing the base case never to be hit and thus continuously

creating copies of the function in an infinite loop, or when the amount of copies created of the function exceeds the limit of the stack and thus exceeding the limits of the stack causing a crash. *Stack-overflow* was not a real danger in this application because only a small limited number of copies of the function were needed to be created in order to dynamically create the *for* loops. This is because the number of variables used in the various test cases was relatively small compared to the size of the instruction stack. The reason the number of variables used was relatively small is because the variables represent a limited number of physical parameters.

3.5 Pseudo-code representation

Shown below is a pseudo-code representation of the recursive *for* loop implementation:

for $k_0 = 0$ to p **do**

$g = 0$;

$kays\,[g] = k_0$

doFor (k_0)

end for

Now the code above makes a function call to the function doFor() which is shown in pseudo-code below:

doFor (k_u)

$g++$

for $k = 0$ to $p - kays\,[g]$ **do**

if $g == G$ **then**

do some processing

else

doFor (k)

end if

end for

end doFor

Note that the variable G represents the number of variables which includes the complex frequency variable and that it starts from zero.

3.6 File I\O

Information needed by the system is loaded in *.mat* files. In order to not have to rewrite the program that reads this file every time the number of variables changed, a standardization was developed for the input file. In the input data file each column represents a certain aspect of the inputted variable data, while each row represents one individual data vector and therefore the number of rows equals the number of data vectors given to the application. Table 3–1 shows the general format for the input data file. Shown in Table 3–1 is the input data file for $G = 2$ meaning that there are three input variables with one variable being the complex frequency variable and the other two variables being the geometric variables. Also shown is $n = 1$ meaning that the highest derivative term is of the first order. This input file format can be extended for more geometric variables and for higher-order derivatives by simply following the pattern shown below in Table 3–1 :

Table 3–1: Input data file configuration

x_0	x_1	x_2	H	$\frac{\partial}{\partial x_1}H$	$\frac{\partial}{\partial x_2}H$	$\frac{\partial}{\partial x_0}H$	$\frac{\partial}{\partial x_1}\left(\frac{\partial}{\partial x_0}H\right)$	$\frac{\partial}{\partial x_2}\left(\frac{\partial}{\partial x_0}H\right)$

CHAPTER 4
Results and Analysis

The following chapter will present the results obtained from various different test cases. The first set of results presented is for the RLC test case, followed by the results based on data obtained using the Finite Element Method.

4.1 RLC Test Case

The RLC test case refers to the simulation of a circuit containing a resistor, an inductor and a capacitor. The circuit shown in figure 4–1 functions as a bandpass filter [15]. The goal of this test case is to use cauchy interpolation to reproduce the rational polynomial $H = \frac{V_o}{V_i}$.

The RLC circuit in figure 4–1 is a good test model for the cauchy interpolation since in this example, the exact rational polynomial of the transfer function is known. The transfer function H can be written as:

$$H = \frac{V_o}{V_i} = \frac{\omega^2 LC + 1}{j\omega CR - \omega^2 LC + 1} \tag{4.1}$$

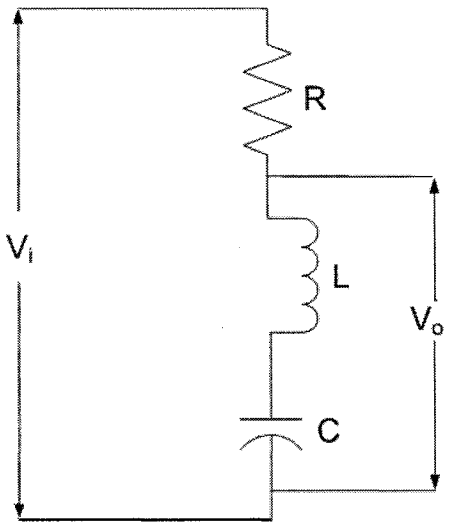


Figure 4–1: RLC Circuit

The variables R, L ,C and ω can be replaced by x_0, x_1, x_2 and x_3. ω is the variable for the input frequency of the signal given to the system. By varying the values of R, L, C and ω a multi-variate case can be generated. Then taking the derivatives at each of these sets of points gives a multi-variate, multi-derivative test case. Shown in figures 4–2 through 4–5 are the results of Cauchy interpolation for the RLC test case. The RLC test case was generated using 4 distinct values for R, L and C, and then taking 4 distinct values of ω based on the resonant point of the current L and C values as described in the pseudo-code algorithm given below. Furthermore each data point was calculated for n equal to 0 and 1, meaning 1 derivative of the transfer function H with respect to the R, L and C values. Therefore there are there are $4 \times 4 \times 4 \times 4 = 256$ data points (k_0,h,r,u) and 8 values per data point,

giving the number of rows $N = 2048$ in the Cauchy matrix. Below is a pseudo-code representation of the system with the actual values used to generate the data points:

for R $= \{50, 75, 100, 125\}$ **do**

  for L $= \{5 \times 10^{-3}, 8 \times 10^{-3}, 11 \times 10^{-3}, 14 \times 10^{-3}\}$ **do**

    for C $= \{2 \times 10^{-3}, 4 \times 10^{-3}, 6 \times 10^{-3}, 8 \times 10^{-3}\}$ **do**

      $\omega = \dfrac{1}{\sqrt{LC}}$

      for $\omega = \{0.6 \times \omega, 0.8 \times \omega, 1 \times \omega, 1.2 \times \omega\}$ **do**

        The following data values are given at (ω, R, L, C):

        $H, \dfrac{\partial H}{\partial R}, \dfrac{\partial H}{\partial L}, \dfrac{\partial H}{\partial C}$; and $\dfrac{\partial}{\partial \omega}\left(H, \dfrac{\partial H}{\partial R}, \dfrac{\partial H}{\partial L}, \dfrac{\partial H}{\partial C}\right)$

      end for

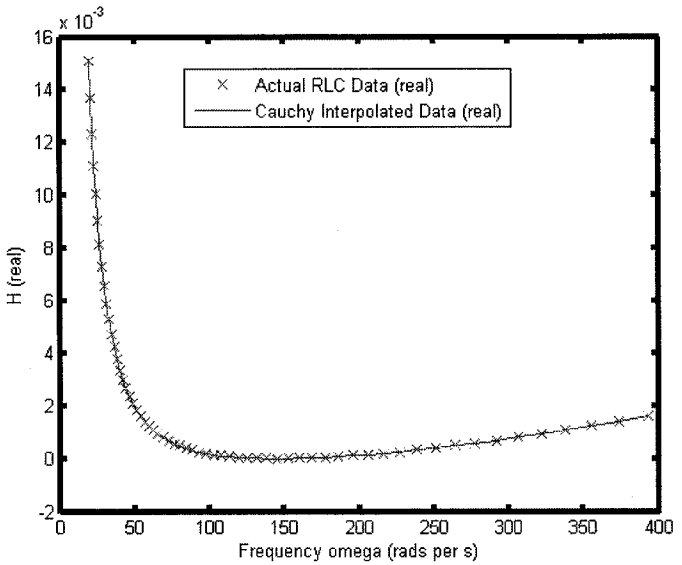
    end for

  end for

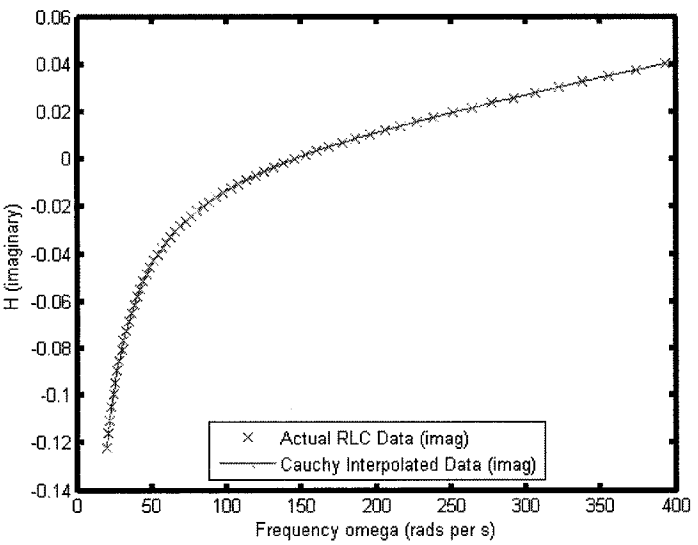
end for

For each variable, a 1-D slice was taken varying just that variable and keeping the other variables constant and the actual transfer function at these points was compared to the transfer function generated using Cauchy Interpolation. Each slice consisted of 120 evenly spaced points through the center point, but with none of the values of the slice containing any of the same points that were used to generate the Cauchy matrix. The results shown on the following pages clearly show that when $p = q = 4$ the Cauchy interpolation can generate a result for the transfer function that is identical to the actual value. Setting $p = q = 4$ gives the exact results because the term $\omega^2 LC$ in equation (4.1)has the greatest sum of exponents of all the terms,

and is equal to 4. The interpolated results in figures 4–2 to 4–5 differ slightly from the exact values in some places, due to numerical round off. In exact arithmetic, the agreement would be exact.

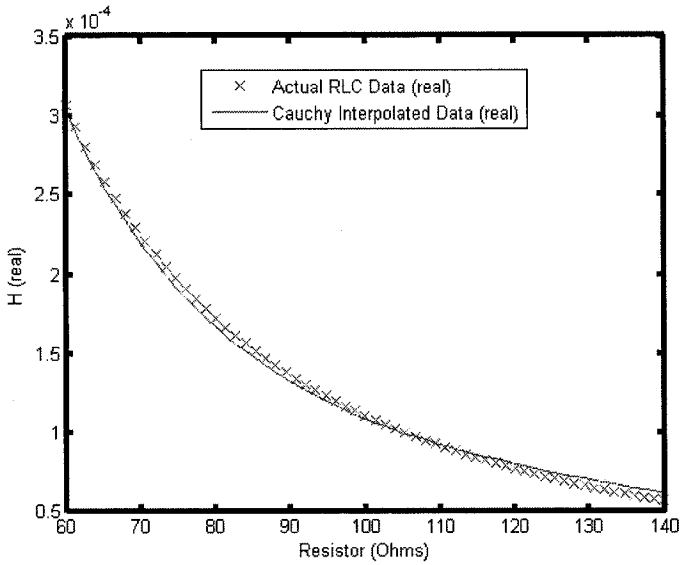


(a) RLC test case for the ω slice using TLS (Real part)

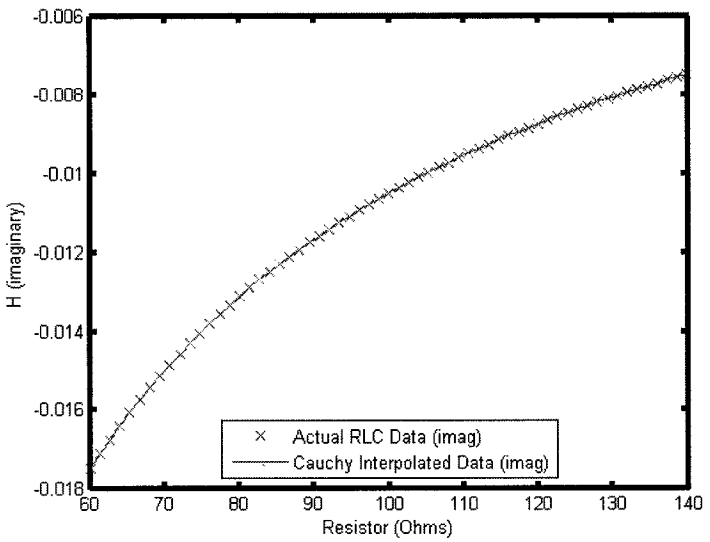


(b) RLC test case for the ω slice using TLS (Imaginary part)

Figure 4–2: RLC test case when $p = q = 4$ for the ω slice using TLS

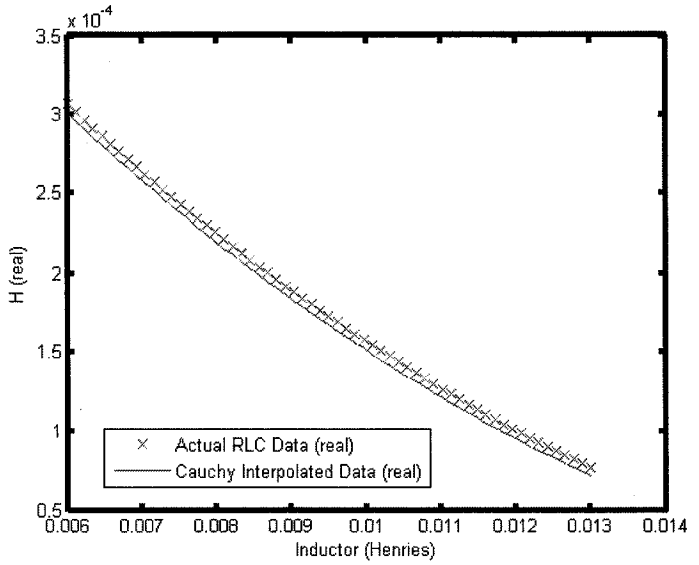


(a) RLC test case for the R slice using TLS (Real part)

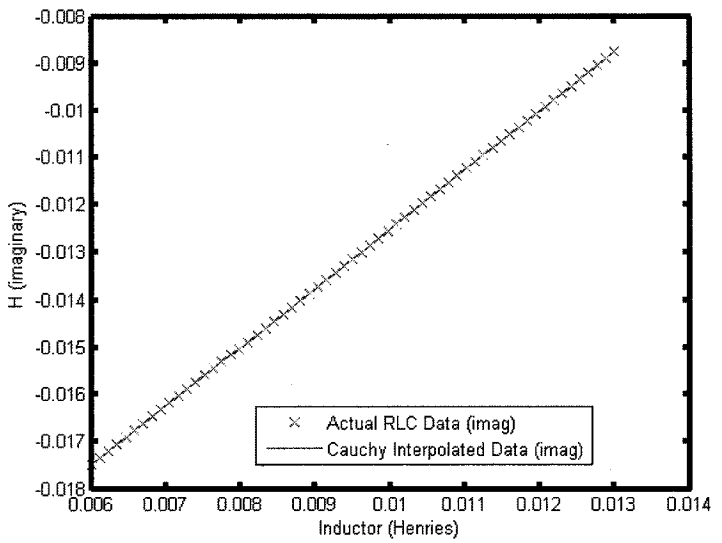


(b) RLC test case for the R slice using TLS (Imaginary part)

Figure 4–3: RLC test case when $p = q = 4$ for the R slice using TLS

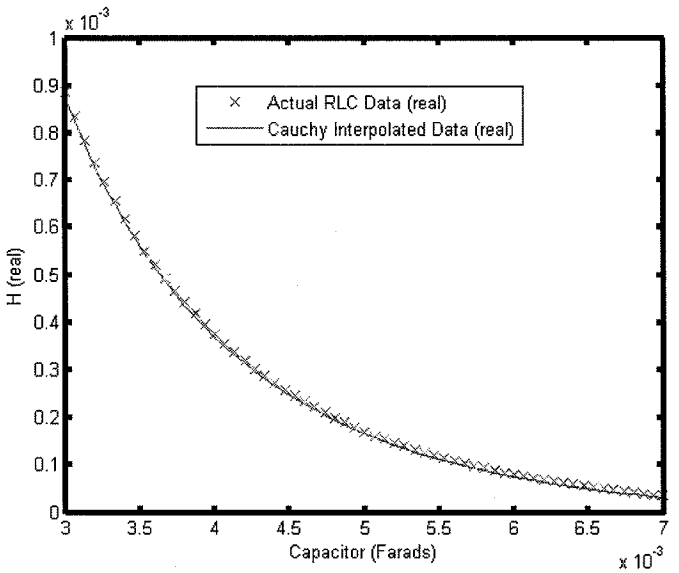


(a) RLC test case for the L slice using TLS (Real part)

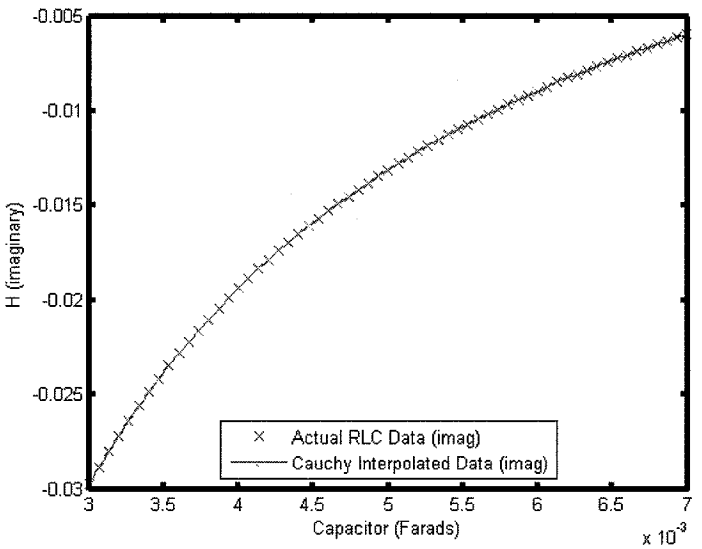


(b) RLC test case for the L slice using TLS (Imaginary part)

Figure 4–4: RLC test case when $p = q = 4$ for the L slice using TLS



(a) RLC test case for the C slice using TLS (Real part)



(b) RLC test case for the C slice using TLS (Imaginary part)

Figure 4–5: RLC test case when $p = q = 4$ for the L slice using TLS

4.2 Finite Element Method Test Cases

The two-port microwave component in figure 4–6 consists of a short length of a metal-walled, air-filled rectangular waveguide, in which is located an adjustable metal post. By varying the radius, height and position of the post we can vary the transfer function. The transfer function that was modeled was the complex scattering parameter S_{12}, indicating transmission of the dominant mode of the waveguide from port 1 to port 2. This waveguide is used for the single point multi-derivative test case and the multi-point multi-variate multi-derivative test case. The microwave component model provides for 4 variables, k_0 which is the free-space wavenumber in *rads* m^{-1} and is obtained by dividing ω (the frequency in *rads* s^{-1}) by c (the velocity of light), r the radius of the post in meters, h the height of the post in meters, and u the distance of the post from the wall of the waveguide in meters. The waveguide cross section was fixed at $a = 2m$ and $b = 1m$ (convenient dimensions for calculation purposes that can easily be scaled to realistic values). Presented in Table 4–1 are the minimum and maximum values of each of the variables used.

Table 4–1: Minimum and maximum values of the parameters for the waveguide model

Variable	Minimum Value	Maximum Value
k_0	1.72	2.82
h	0.1	0.9
r	0.05	0.15
u	0.25	1.75

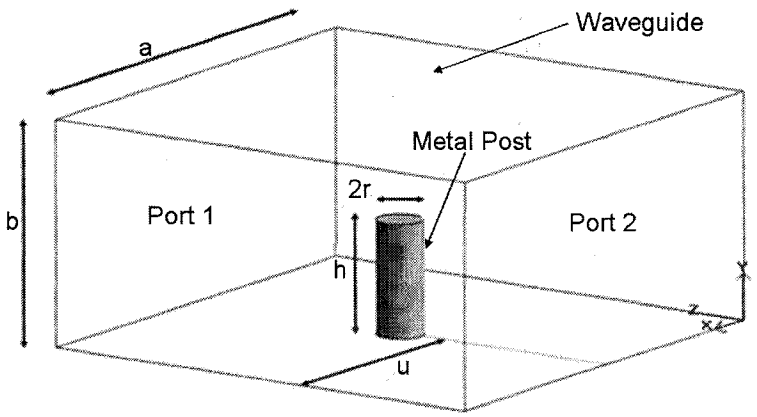


Figure 4–6: Waveguide model

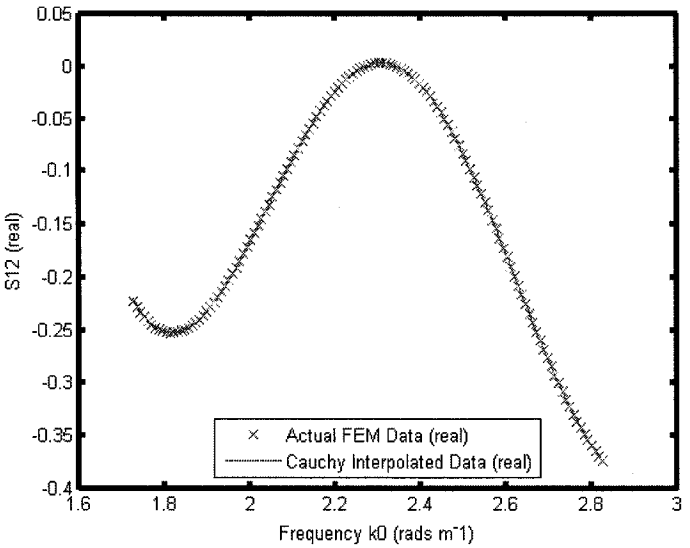
4.2.1 Obtaining the FE Data

The finite element results used for the test cases were obtained with a 3D code using vector, tetrahedral elements [16]. The code is able to compute scattering parameters of arbitrarily-shaped microwave devices, and to find the sensitivities of those scattering parameters with respect to geometric perturbations (Sensitivities are the first derivative with respect to the geometric parameters) [17] [18]. In addition, it can find high order derivatives with respect to frequency, for both the scattering parameters and their sensitivities, using a Taylor series expansion and Padé approximation [1].

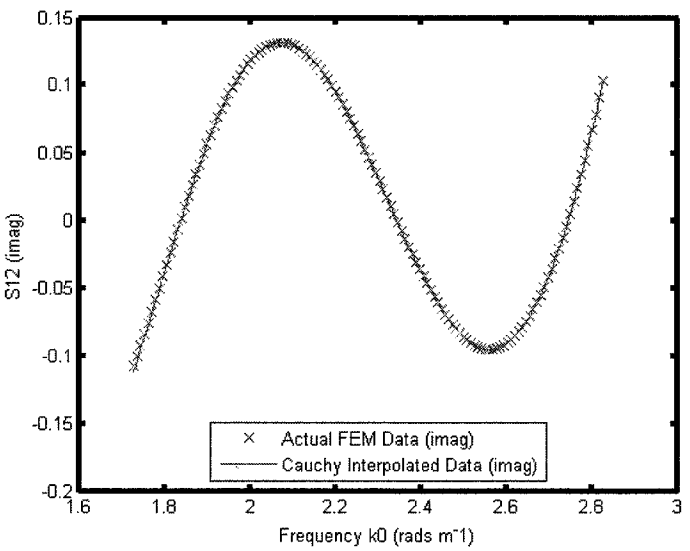
4.2.2 Single Point Multi-Derivative Case

In the single data point multi-derivative case, only one frequency point was input to the system, along with its derivatives. As more higher order derivatives were input

to the system the size of p and q could be increased, and as this happened the Cauchy interpolation results and the actual finite element S_{12} values converge. In figure 4–7 are the results of the single variable case, with $p = q = 8$ and the number of derivatives n of frequency with respect to S_{12} is 20. The values for h, r and u were fixed at 0.75, 0.1 and 1 meters, respectively.



(a) Single point multi-derivative test case using TLS (Real part)



(b) Single point multi-derivative test case using TLS (Imaginary part)

Figure 4–7: Single point multi-derivative test case

4.2.3 The Multi-Variate Multi-Derivative Test Cases

In the multi-variate, multi-derivative test case, the 4 variables k_0, h, r, and u are given to the system. Presented are the results for the least squares (LS) technique for solving the coefficients of the transfer function and for the total least squares (TLS) technique for solving the coefficients of the transfer function.

4.2.4 Results for the 2x2x2x2 Test Case

The 2x2x2x2 test case was generated with 2 different points for each of the four variables, and with n (the derivative of S_{12} with respect to geometry) equal to 0 or 1, meaning either no derivative of S_{12} with respect to any of the geometric variables (h, r and u), or 1 derivative of S_{12} with respect to the geometric variables. Presented below is the algorithm used to generate the 2x2x2x2 test case:

$nPoints = 2$

$$dk_0 = \frac{(k_{0_{high}} - k_{0_{low}})}{nPoints} \text{ and } dh = \frac{(h_{high} - h_{low})}{nPoints}$$

$$dr = \frac{(r_{high} - r_{low})}{nPoints} \text{ and } du = \frac{(u_{high} - u_{low})}{nPoints}$$

for $k_0 = k_{0_{low}} + \frac{dk_0}{2}$ to $k_{0_{high}} - \frac{dk_0}{2}$ increment dk_0 **do**

for $h = h_{low} + \frac{dh}{2}$ to $h_{high} - \frac{dh}{2}$ increment dh **do**

for $r = r_{low} + \frac{dr}{2}$ to $r_{high} - \frac{dr}{2}$ increment dr **do**

for $u = u_{low} + \frac{du}{2}$ to $u_{high} - \frac{du}{2}$ increment du **do**

The following data values are given at (k_0, h, r, u):

$$S_{12}, \frac{\partial S_{12}}{\partial h}, \frac{\partial S_{12}}{\partial r}, \frac{\partial S_{12}}{\partial u}; \text{ and } \frac{\partial}{\partial s}\left(S_{12}, \frac{\partial S_{12}}{\partial h}, \frac{\partial S_{12}}{\partial r}, \frac{\partial S_{12}}{\partial u}\right)$$

end for

end for

end for

end for

For $nPoints = 2$ there are $2 \times 2 \times 2 \times 2 = 16$ data points (k_0,h,r,u) and 8 values per data point, giving the number of rows $N = 128$. Presented in Table 4–2 are the results obtained for the 2x2x2x2 test cases. Given is the RMS error for when the total least squares (TLS) technique is used and for when the least squares (LS) technique is used. The RMS error refers to the root mean square error and it was calculated by taking the magnitude of the difference at each point, then taking the square of this magnitude, summing up all the squares over all the points and then dividing this sum by the number of points and then by taking the square root.

Table 4–2: RMS Errors for the 2x2x2x2 Test Cases

			RMS Error	
p=q	P+Q+2	Rank	TLS	LS
1	10	10	0.57	52.32
2	30	30	0.76	0.95
3	70	70	0.18	0.03

Table 4–3 and 4–4 refers to the RMS error per slice using the least squares (LS) and total least squares method (TLS) respectively. The four slices represent k_0 the

frequency variable and the three variables for the dimensions h, r , u. Each slice goes through the center of the 4D box in parameter space, whose dimensions are defined by Table 4–1. Each slice consists of 129 equally spaced points. It is also important to note that each slice contains none of the original data points used to build the interpolation. After the transfer function is generated using cauchy interpolation, you can vary one of these parameters while holding the other three steady to see the results of varying one variable on the transfer function. The results obtained for the RMS error for the four slices using the least square method (LS) and total least squares are shown in Table 4–3 and Table 4–4 respectively. N is the number of rows in the matrix and the rank was as the number of non-zero values from the singular value decomposition of the matrix.

Table 4–3: RMS Errors Per Slice for the 2x2x2x2 Test Cases Using LS

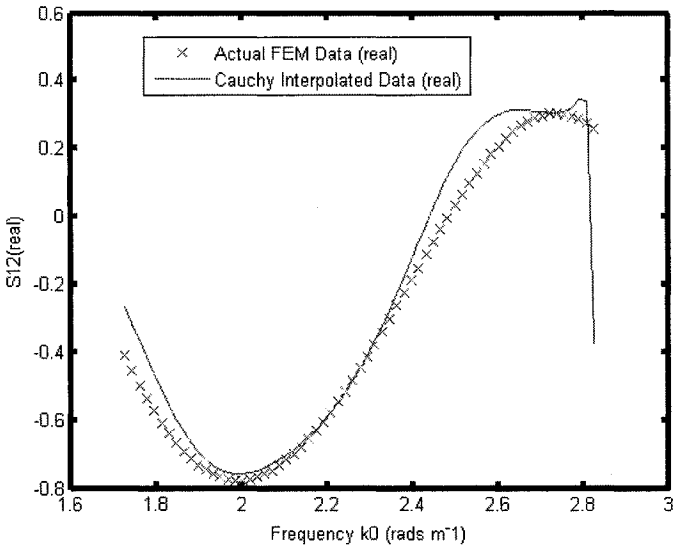
				RMS Error per Slice (LS)			
p=q	N	P+Q+2	Rank	f	h	r	u
1	128	10	10	91.94	108.92	86.16	88.54
2	128	30	30	1.38	1.73	1.32	1.59
3	128	70	70	0.50	0.20	0.08	20.80

Shown in Table 4–4 are the results obtained for the RMS error of the four slices using the total least squares method (TLS).

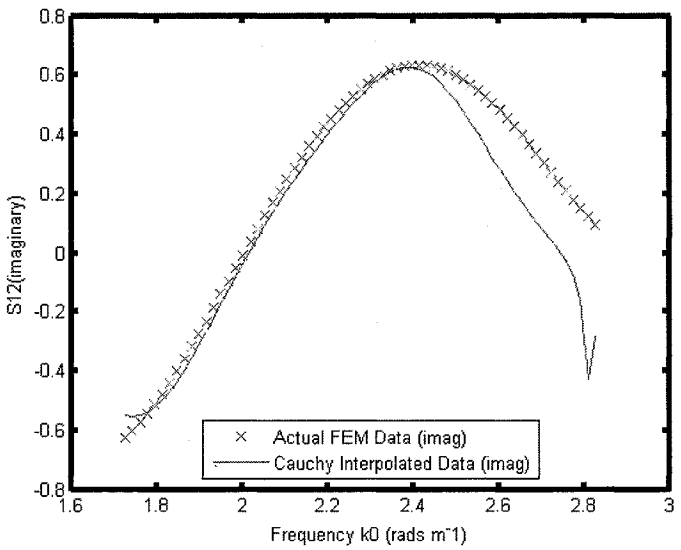
Table 4–4: RMS Errors Per Slice for the 2x2x2x2 Test Cases Using TLS

				RMS Error per Slice (TLS)			
p=q	N	P+Q+2	Rank	f	h	r	u
1	128	10	10	1.23	1.70	1.86	1.84
2	128	30	30	1.39	1.54	1.17	1.61
3	128	70	70	1.41	0.98	1.15	1.23

Shown in figure 4–8 through figure 4–11 are the graphs obtained for the 2x2x2x2 test case when $p = q = 3$ using the Least Squares method. In figures 4–11(a) and 4–11(b) the two spikes are caused by a large numerator over a small denominator. This hides the results in the center of the graph, in figures 4–12(a) and 4–12(b) are zoomed views of the centers of those graphs. Shown in figure 4–13 through 4–16 are the 2x2x2x2 test case with $p = q = 3$ when the Total Least Squares method is used.

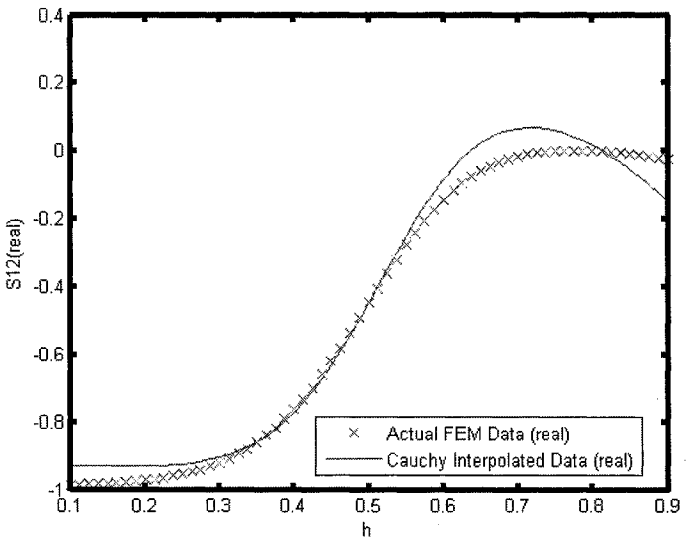


(a) 2x2x2x2 test case for *frequency* slice using LS (Real Part)

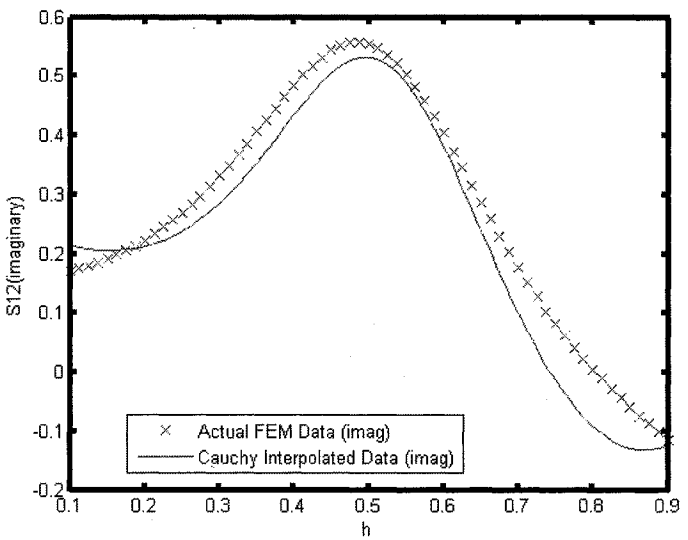


(b) 2x2x2x2 test case for *frequency* slice using LS (Imaginary part)

Figure 4–8: 2x2x2x2 test case when $p = q = 3$ for *frequency* slice using LS

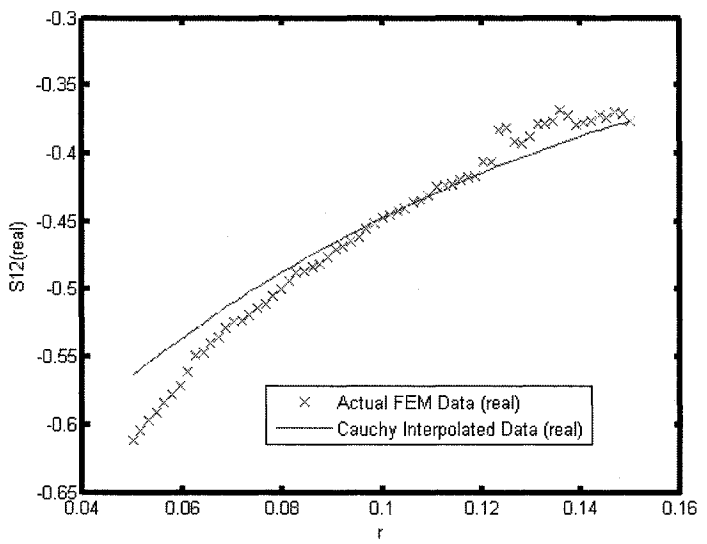


(a) 2x2x2x2 test case for h slice using LS (Real part)

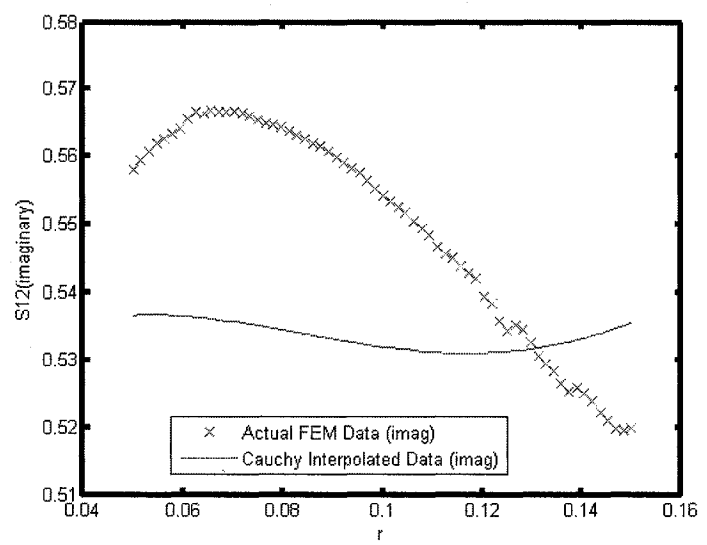


(b) 2x2x2x2 test case for h slice using LS (Imaginary part)

Figure 4–9: 2x2x2x2 test case when $p = q = 3$ for h slice using LS

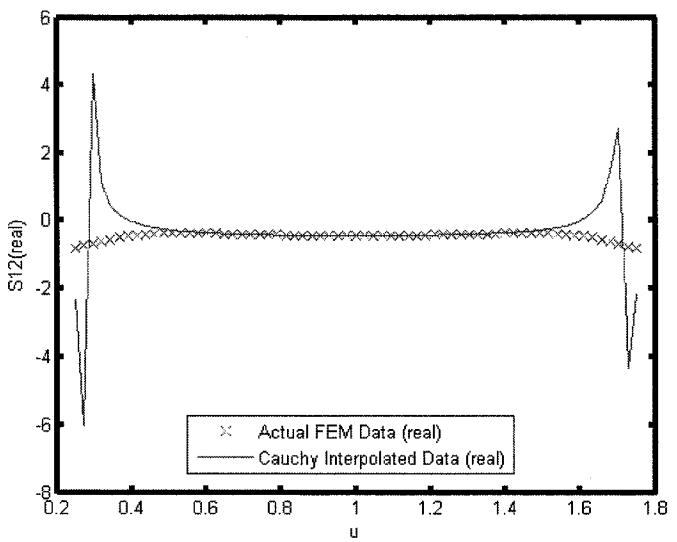


(a) 2x2x2x2 test case for r slice using LS (Real part)

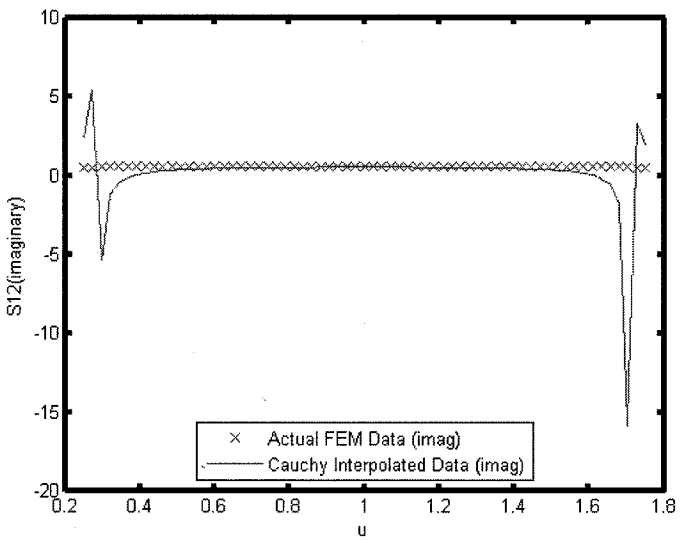


(b) 2x2x2x2 test case for r slice using LS (Imaginary part)

Figure 4–10: 2x2x2x2 test case when $p = q = 3$ for r slice using LS

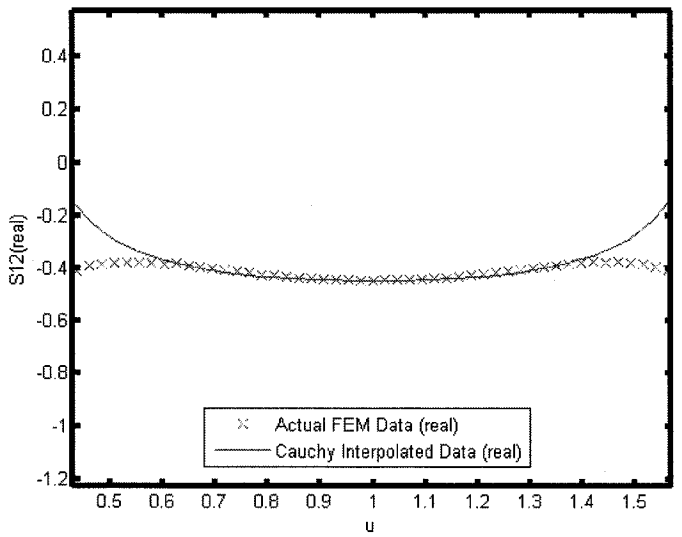


(a) 2x2x2x2 test case for u slice using LS (Real part)

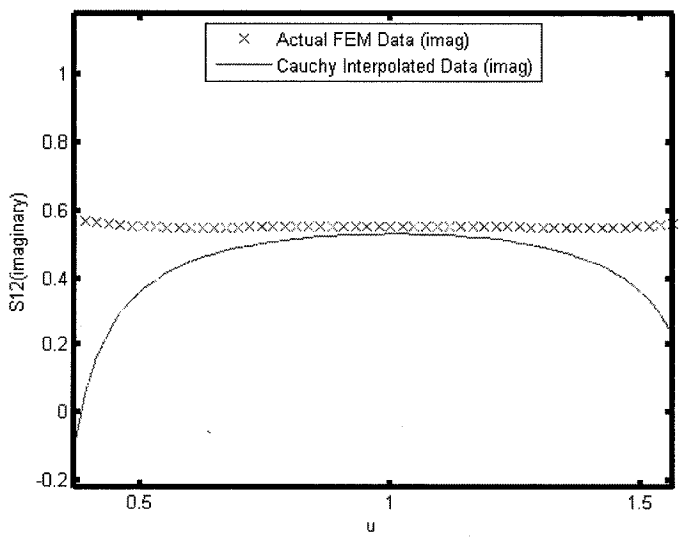


(b) 2x2x2x2 test case for u slice using LS (Imaginary part)

Figure 4–11: 2x2x2x2 test case when $p = q = 3$ for u slice using LS

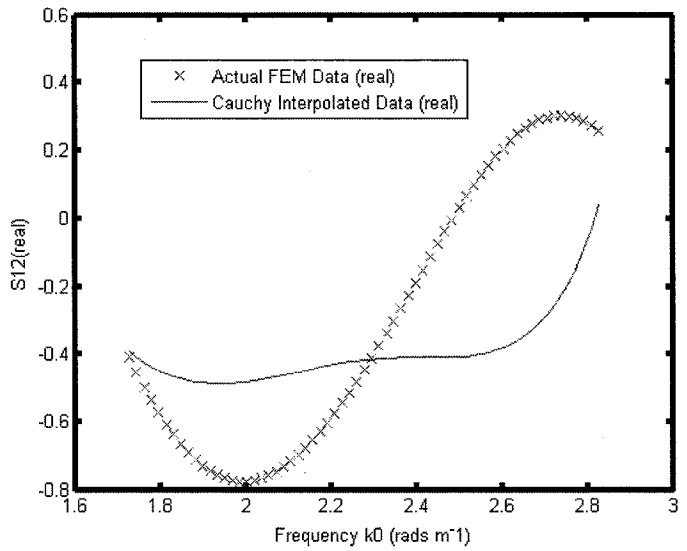


(a) 2x2x2x2 magnified results for test case for u slice using LS (Real part)

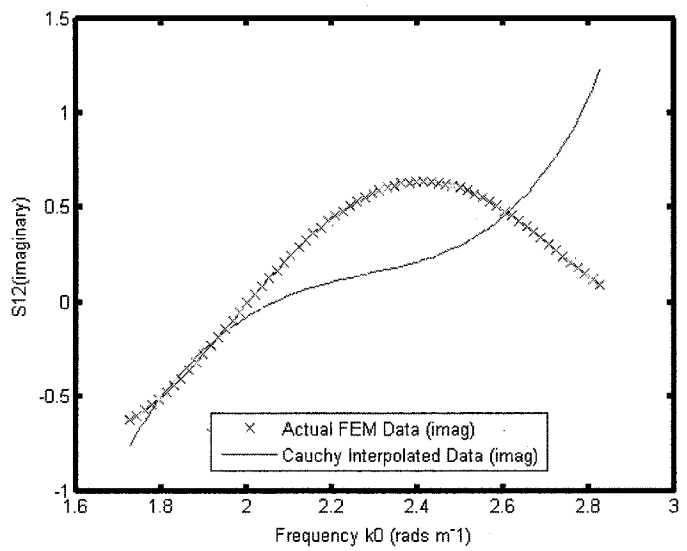


(b) 2x2x2x2 magnified results for test case for u slice using LS (Imaginary part)

Figure 4–12: 2x2x2x2 magnified results for test case when $p = q = 3$ for u slice using LS

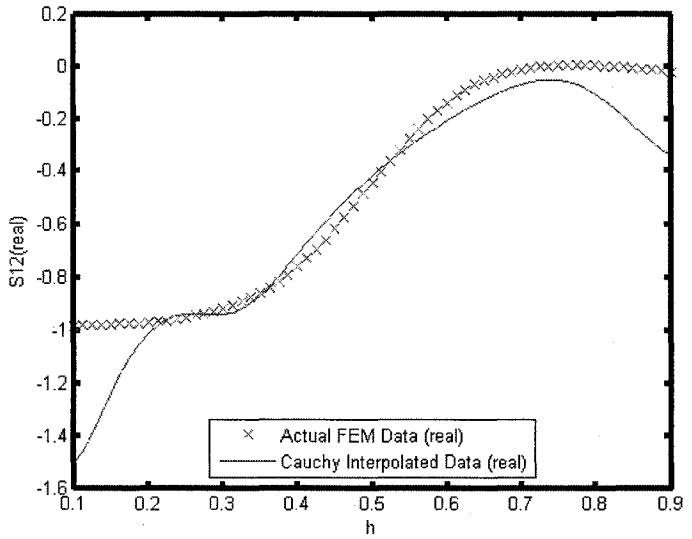


(a) 2x2x2x2 test case for *frequency* slice using TLS (Real part)

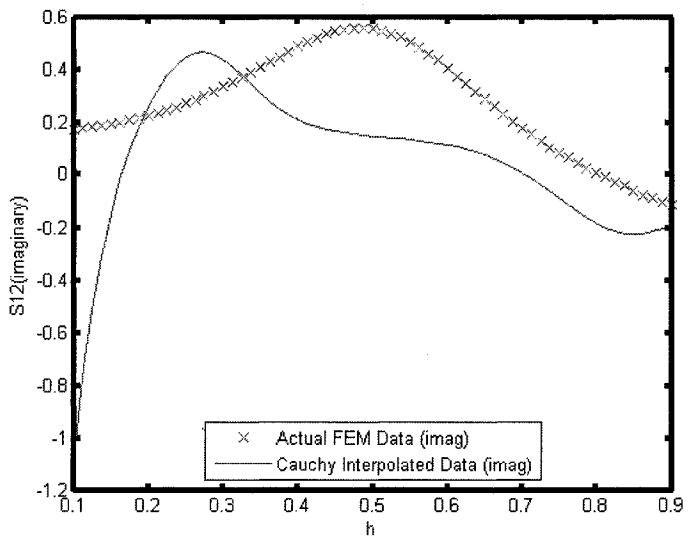


(b) 2x2x2x2 test case for *frequency* slice using TLS (Imaginary part)

Figure 4–13: 2x2x2x2 test case when $p = q = 3$ for *frequency* slice using TLS

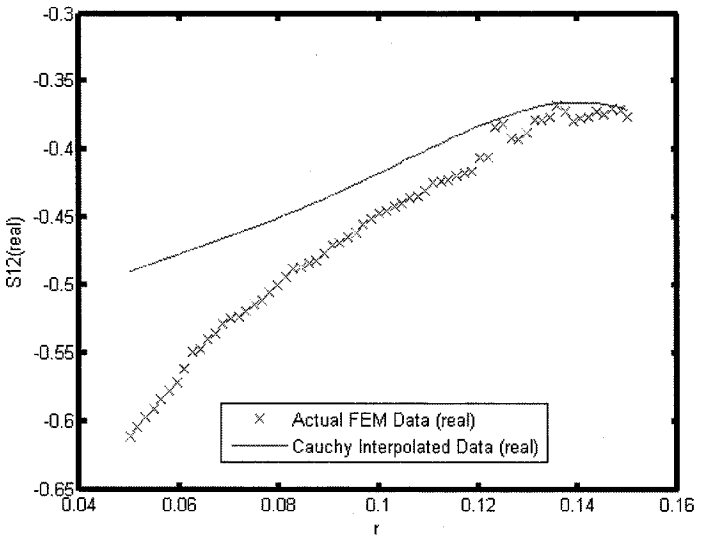


(a) 2x2x2x2 test case for h slice using TLS (Real part)

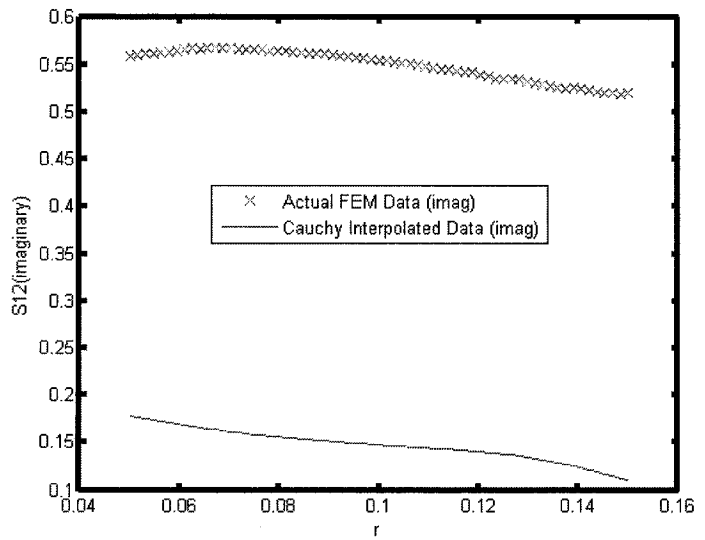


(b) 2x2x2x2 test case for h slice using TLS (Imaginary part)

Figure 4–14: 2x2x2x2 test case when $p = q = 3$ for h slice using TLS

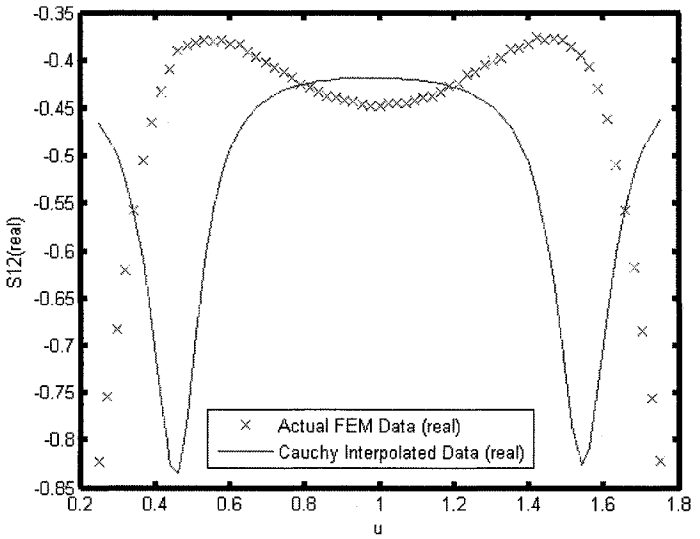


(a) 2x2x2x2 test case for r slice using TLS (Real part)

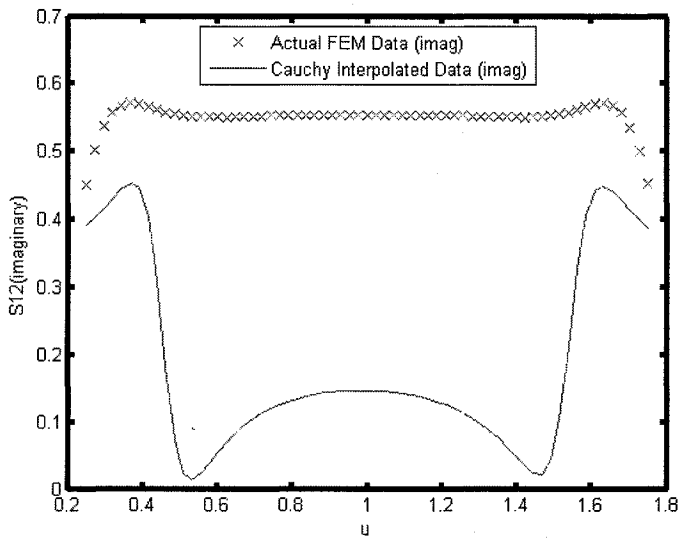


(b) 2x2x2x2 test case for r slice using TLS (Imaginary part)

Figure 4–15: 2x2x2x2 test case when $p = q = 3$ for r slice using TLS



(a) 2x2x2x2 test case for u slice using TLS (Real part)



(b) 2x2x2x2 test case for u slice using TLS (Imaginary part)

Figure 4–16: 2x2x2x2 test case when $p = q = 3$ for u slice using TLS

4.2.5 Results for the 4x4x4x4 Test Case

The 4x4x4x4 test case is similar to the 2x2x2x2 case but here the test case was generated with 4 different points for each of the four variables, and n (the derivative of S_{12} with respect to geometry) equal to 0 or 1, meaning either no derivative of S_{12} with respect to any of the geometric variables (h, r and u), or 1 derivative of S_{12} with respect to the geometric variables. The 4x4x4x4 case uses the same algorithim presented in section 4.2.4 but in this case $nPoints = 4$, there are $4 \times 4 \times 4 \times 4 = 256$ data points (k_0,h,r,u) and 8 values per data point, giving the number of rows $N = 2048$.

Presented in Table 4–5 are the results obtained for the 4x4x4x4 test cases. Given are the RMS errors for when the total least squares technique and for when the least squares technique are used. N is the number of rows in the matrix. The rank was computed as the number of non-zero values of the singular decomposition of the matrix. Also given is the condition number of the Cauchy matrix. The condition number of a matrix is the ratio of the largest singular value of the matrix and the smallest. A large condition number (typically $> 10^{12}$) indicates that the matrix is nearly singular.

Table 4–5: RMS Errors for the 4x4x4x4 Test Cases

					RMS Error	
p=q	P+Q+2	N	Rank	Condition Number	TLS	LS
4	140	2048	140	1.10×10^{8}	0.18	1.04
5	252	2048	252	7.90×10^{11}	0.05	0.14
6	420	2048	417	7.70×10^{13}	0.07	0.04
7	660	2048	638	1.40×10^{16}	0.03	0.05

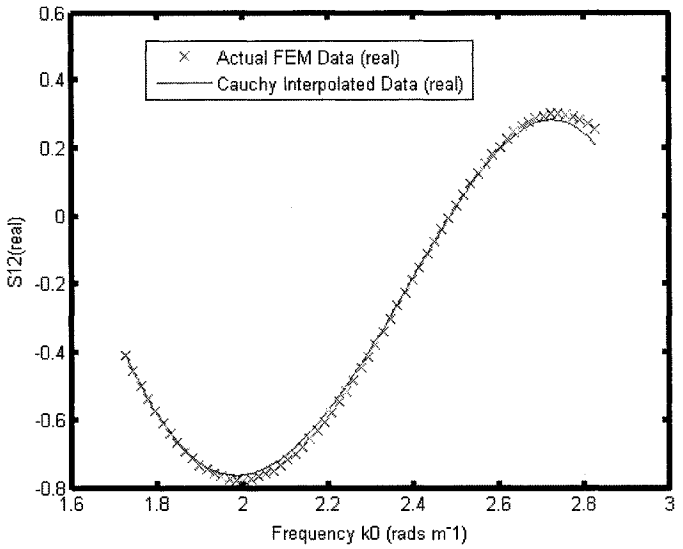
Table 4–6 lists the RMS error per each of the 4 slices: frequency, h, r, u.

Table 4–6: RMS Errors Per Slice for the 4x4x4x4 Test Cases

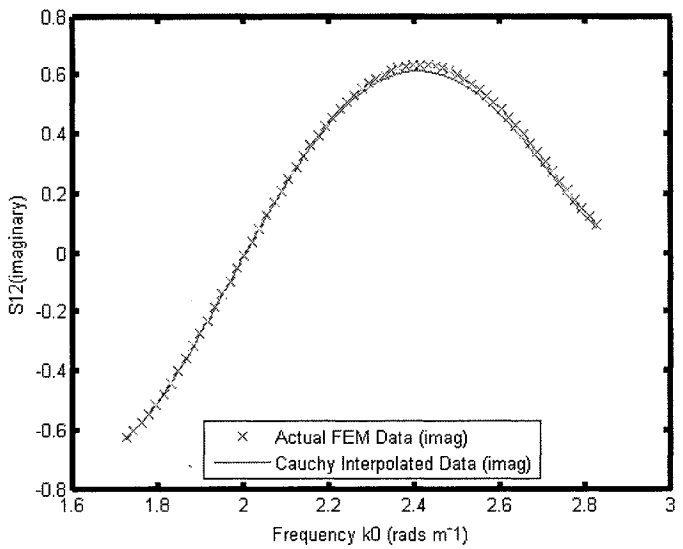
			RMS Error per Slice (TLS)				RMS Error per Slice (LS)			
p=q	P+Q+2	Rank	f	h	r	u	f	h	r	u
4	140	140	0.06	0.06	0.05	0.10	1.66	0.67	0.50	1.34
5	252	252	0.03	0.02	0.04	0.04	0.05	0.08	0.02	0.10
6	417	417	0.15	0.05	0.06	0.05	0.01	0.05	0.03	0.03

Shown in figure 4–17 through figure 4–20 are the graphs obtained for the 4x4x4x4 test case when $p = q = 6$ using the least squares method. These are the best results obtained for the 4x4x4x4 case using the least squares method.

Shown in figure 4–21 through figure 4–24 are the graphs obtained for the 4x4x4x4 test case when $p = q = 5$ using the total least squares method. These are the best results obtained for the 4x4x4x4 case using the total least squares method.

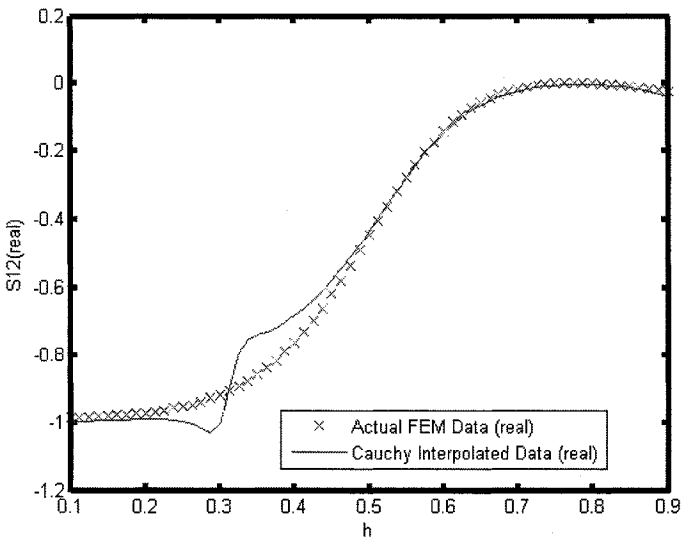


(a) 4x4x4x4 test case for *frequency* slice using LS (Real part)

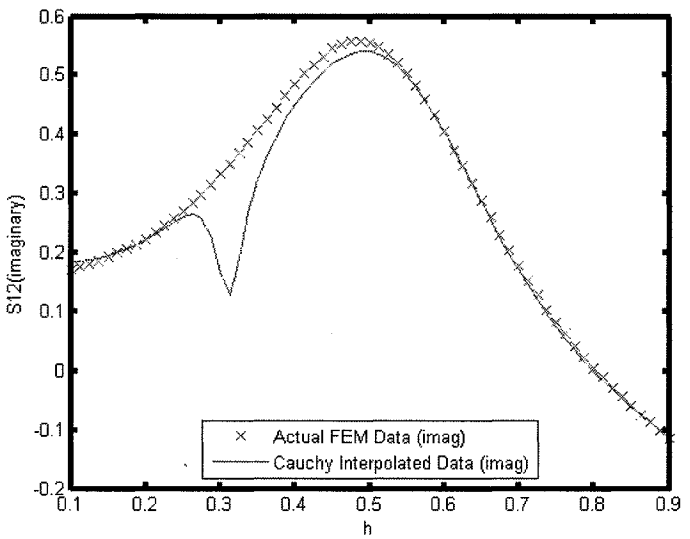


(b) 4x4x4x4 test case for *frequency* slice using LS (Imaginary part)

Figure 4–17: 4x4x4x4 test case when $p = q = 6$ for *frequency* slice using LS

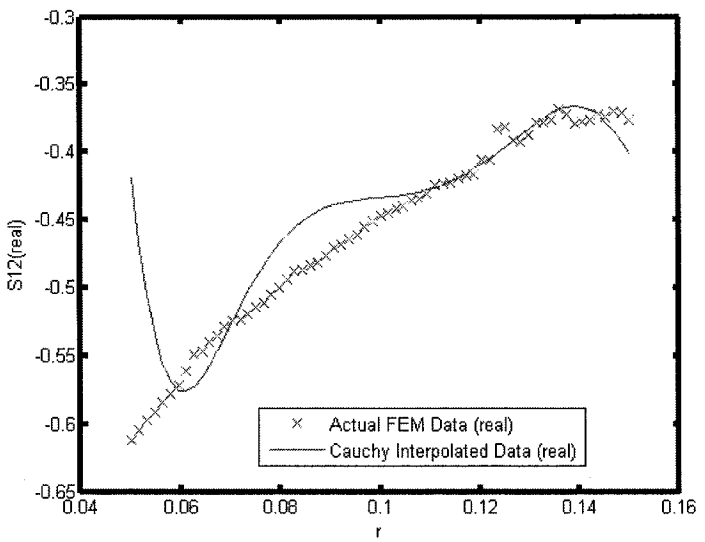


(a) 4x4x4x4 test case for h slice using LS (Real part)

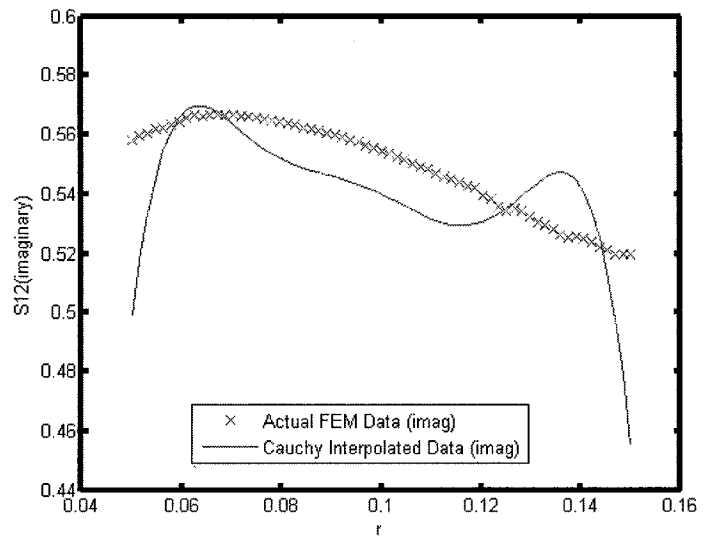


(b) 4x4x4x4 test case for h slice using LS (Imaginary part)

Figure 4–18: 4x4x4x4 test case when $p = q = 6$ for h slice using LS

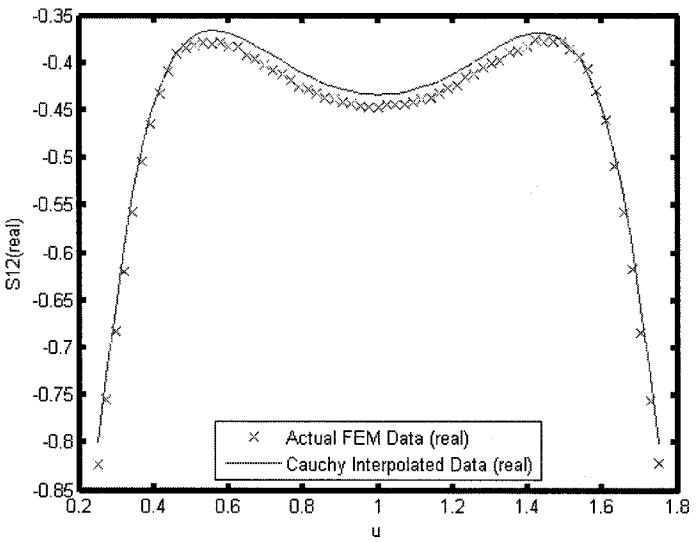


(a) 4x4x4x4 test case for r slice using LS (Real part)

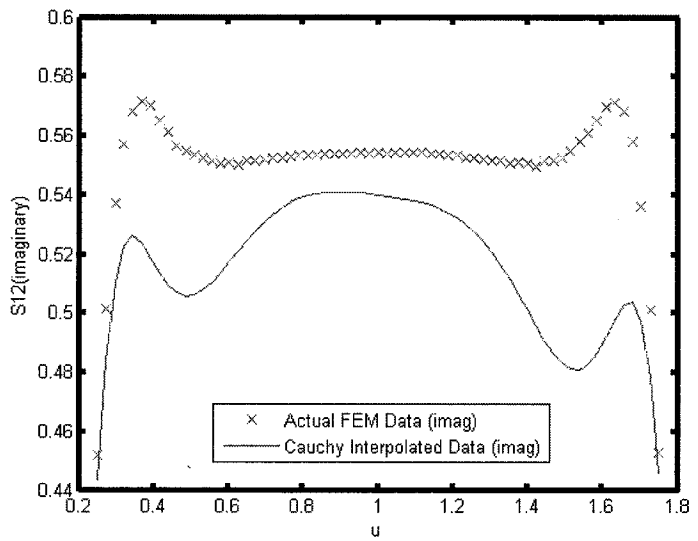


(b) 4x4x4x4 test case for r slice using LS (Imaginary part)

Figure 4–19: 4x4x4x4 test case when $p = q = 6$ for r slice using LS

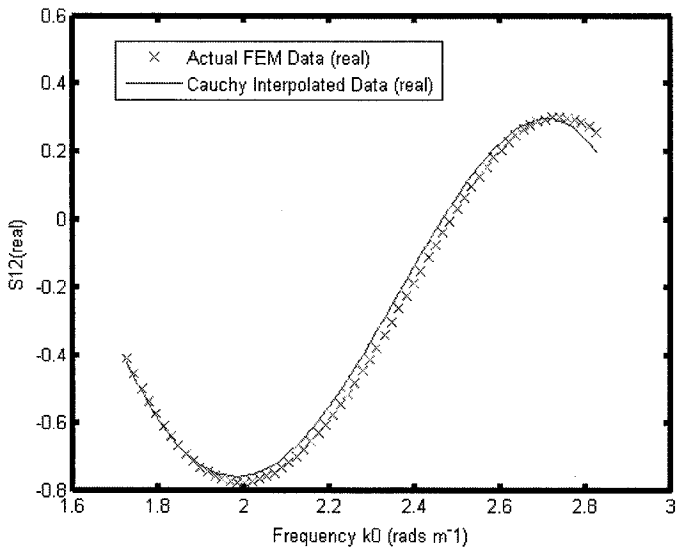


(a) 4x4x4x4 test case for u slice using LS (Real part)

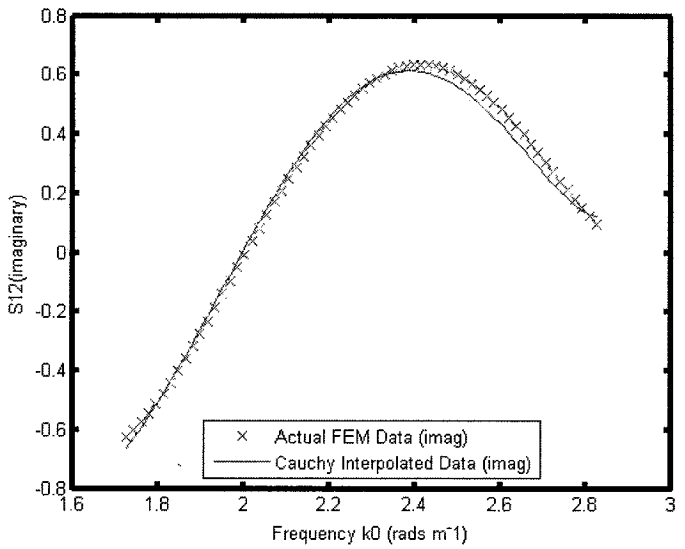


(b) 4x4x4x4 test case for u slice using LS (Imaginary part)

Figure 4–20: 4x4x4x4 test case when $p = q = 6$ for u slice using LS

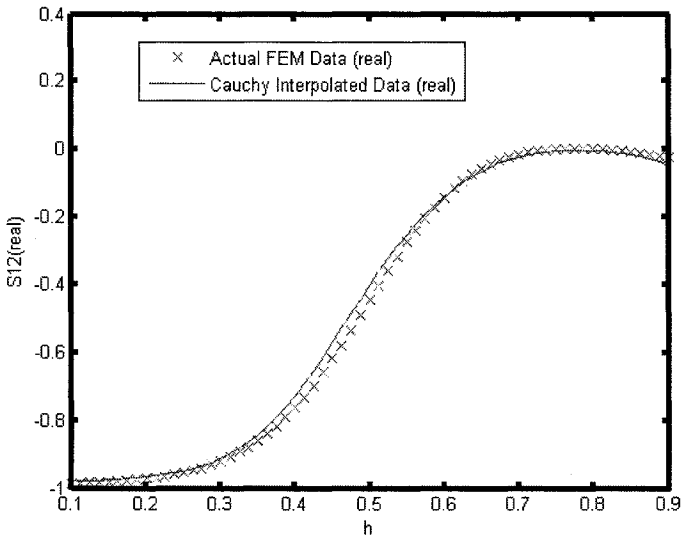


(a) 4x4x4x4 test case for *frequency* slice using TLS (Real part)

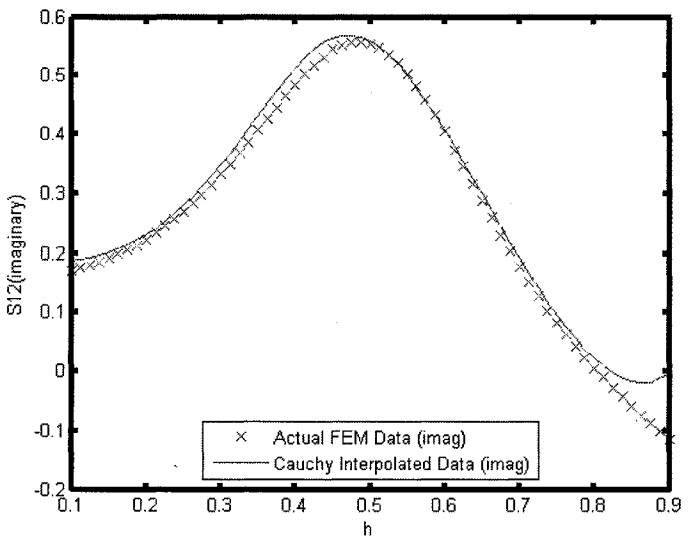


(b) 4x4x4x4 test case for *frequency* slice using TLS (Imaginary part)

Figure 4–21: 4x4x4x4 test case when $p = q = 5$ for *frequency* slice using TLS

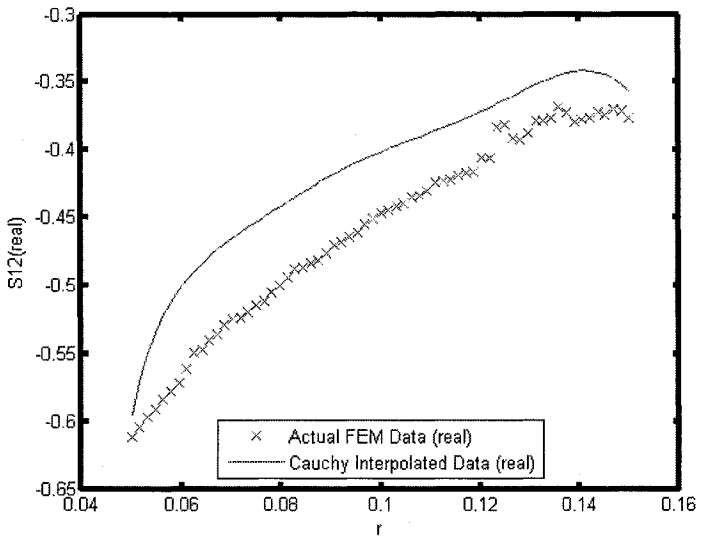


(a) 4x4x4x4 test case for h slice using TLS (Real part)

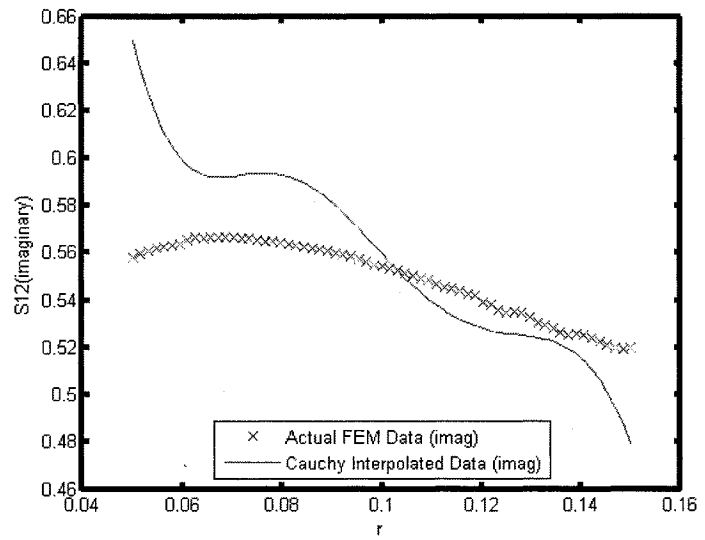


(b) 4x4x4x4 test case for h slice using TLS (Imaginary part)

Figure 4–22: 4x4x4x4 test case when $p = q = 5$ for h slice using TLS

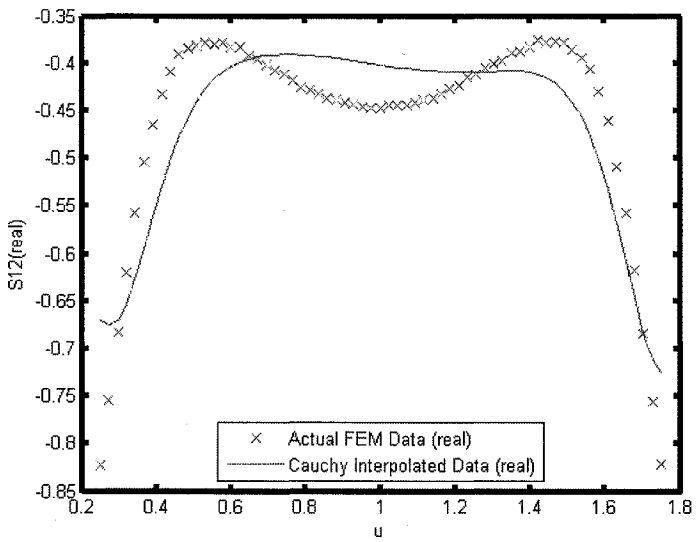


(a) 4x4x4x4 test case for r slice using TLS (Real part)

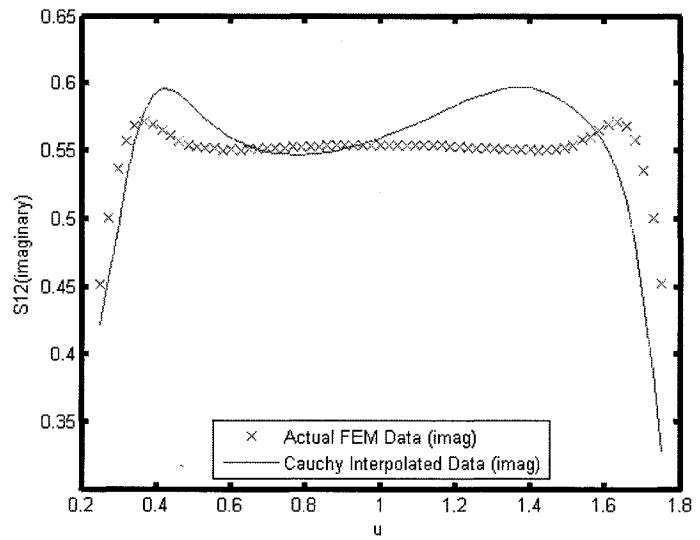


(b) 4x4x4x4 test case for r slice using TLS (Imaginary part)

Figure 4–23: 4x4x4x4 test case when $p = q = 5$ for r slice using TLS



(a) 4x4x4x4 test case for u slice using TLS (Real part)



(b) 4x4x4x4 test case for u slice using TLS (Imaginary part)

Figure 4–24: 4x4x4x4 test case when $p = q = 5$ for u slice using TLS

4.2.6 Discussion of Results

The results given show that using total least squares may be ignoring by virtue of its use of the error matrix some features of the data that lead the least squares (LS) method to generate interpolation results that seems to have some wild oscillations in some regions. These wild oscillations give rise to a few slice points that have very high errors, as can be seen in some of the figures for the LS test cases. The TLS method therefore is preferable to the least squares method because although it does not always give as small errors as least squares does, it is more robust, and more tolerant of noise in the data. Therefore it can be concluded from the results presented this in thesis that total least squares is the preferred method for use in the Cauchy interpolation method.

CHAPTER 5
Conclusion

The goal of this thesis was to implement Cauchy interpolation for multi-variate, multi-derivative data. This goal was achieved, as demonstrated in the preceding chapter. This means that cauchy interpolation is an effective means of constructing a rational polynomial for multi-variate multi-derivative test cases.

5.1 Future Work

Future work for this thesis would be the use of adaptive interpolation algorithms. Some possible ideas for an adaptive algorithm is to feed the system a set of data points run the program, and calculate the RMS error, and then if it's not below a certain goal, use a genetic algorithm to change the points and re-run the system until the points that are generated are those that would produce the minimum RMS value. This would allow for the use of the best data points to generate the optimum model of the device. Another idea, is to add points in regions where necessary and remove points in regions where not. The system would run and would add more data points in a region to meet a certain accuracy threshold, if adding additional points doesn't increase accuracy after a certain pre-determined number of data points are added, the system would stop adding points in that region and move on to the next region where it would again start adding points to see if an increase in accuracy is possible by adding data points in that region, this would allow for more data in regions where

more data points is necessary to achieve a higher degree of accuracy and less data in regions where more points aren't needed.

References

[1] J. Webb, "Design sensitivity of frequency response in 3-d finite-element analysis of microwave devices," *IEEE Transactions On Magnetics*, vol. 38, no. 2, pp. 1109–1112, 2002.

[2] R. S. Adve, T. K. Sarkar, S. M. Rao, E. K. Miller, and D. R. Pflug, "Application of the cauchy method for extrapolating/interpolating narrow-band system responses," *IEEE Transactions On Microwave Theory And Techniques*, vol. 45, no. 5, pp. 837–845, 1997.

[3] R. Sanaie, E. Chiprout, M. S. Nakhla, and Q. jun Zhang, "A fast method for frequency and time domain simulation of high-speed vlsi interconnects," *IEEE Transactions On Magnetics*, vol. 42, no. 12, pp. 2562–2570, 1994.

[4] M. Celik, O. Ocali, M. A. Tan, and A. Atalar, "Pole-zero computation in microwave circuits using multipoint padé approximation," *IEEE Transactions On Circuits And Systems-I:Fundamental Theory And Applications*, vol. 42, no. 1, pp. 6–13, 1995.

[5] R. S. Adve and T. K. Sarkar, "The effect of noise in the data on the cauchy method," *MIicrowave And Optical Technology Letters*, vol. 7, no. 5, pp. 242–247, 1994.

[6] K. Kottapalli, T. K. Sarkar, Y. Hua, E. K. Miller, and G. J. Burke, "Accurate computation of wide-band response of electromagnetic systems utilizing narrow-band information," *IEEE Transactions On Microwave Theory And Techniques*, vol. 39, no. 4, pp. 682–687, 1991.

[7] R. Lehmensiek and P. Meyer, "Creating accurate multivariate rational interpolation models of microwave circuits by using efficient adaptive sampling to minimize the number of computational electromagnetic analyses," *IEEE On Microwave Theory And Techniques*, vol. 49, no. 8, pp. 1419–1430, 2001.

[8] A. Lamecki, P. Kozakowski, and M. Mrozowski, "Efficient implementation of the cauchy method for automated cad-model construction," *IEEE Microwave And Wireless Components Letters*, vol. 13, no. 7, pp. 268–270, 2003.

[9] G. Golub and C. Van Loan, *Matrix Computations, 3rd Edition.* Baltimore, MD: The John Hopkins University Press, 1996.

[10] E. Weisstein, *Singular Value Decomposition.* Mathworld, 2006.

[11] H. Anton, *Elementary Linear Algebra.* Danvers, MA: John Wiley & Sons, 1994.

[12] S. V. Huffel and J. Vandewalle, *The Total Least Squares Problem: Computational Aspects and Analysis.* Philadelphia, PA: Society for Industrial and Applied Mathematics, 1991.

[13] G. Strang, *Introductions To Applied Mathematics.* Wellesley, MA: Wellesley-Cambridge Press, 1986.

[14] C. Van Loan, *Introductions To Scientific Computing.* Upper Saddle River, NJ: Prentice Hall, 1997.

[15] J. D. Irwin and C.-H. Wu, *Basic Engineerng Circuit Analysis.* Danvers, MA: John Wiley & Sons, 1998.

[16] J. Webb, "Hierachal vector basis functions of arbitrary order for triangular and tetrahedral finite elements," *IEEE Transactions On Antennas And Propagation*, vol. 47, pp. 1244–1253, 1999.

[17] H. Akel and J. Webb, "Design sensitivities for scattering-matrix calculation with tetrahedral edge elements," *IEEE Transactions On Magnetics*, vol. 36, pp. 1043–1046, 2000.

[18] J. Webb, "Design sensitivities using high-order tetrahedral vector elements," *IEEE Transactions On Magnetics*, vol. 37, pp. 3600–3603, 2001.